FW1 INSPECT Script language reference

By Lubomir Nistor(a)Security-Gurus.de

Contents

Introduction

This reference book should give you an overall list of Checkpoint firewall 1 V4.1 features and functions defined. Although this information is a part of each firewall deployment there was no information published about it and I hope Checkpoint didn't want to cash on this to show that their support pays off..

Although Inspect scripting is not a very commonly used amongst CP firewall engineers or administrators it is a very powerful tool for either modifying the standard behavior of the firewall engine but also for remote modification of the policy in case there is no access to the GUI console..

The information in this book is licensed by Checkpoint and you can use it only if you've bought their products (that's how I understood it), so please buy their products before you go on reading this (lol).

I hope you enjoy using this reference book and gain deeper understanding how the CP engine works (although I'm not quite sure if it does as it is written here).

Firewall engine system

INSPECT is an object-oriented, high-level script language that specifies packet handling by classifying packet content and state. The INSPECT engine examines virtually all communications taking place at and above the network level, and efficiently extracts the relevant data from each packet.
A Security Policy is defined using VPN-1/FireWall-1's graphical user interface. From the Security Policy, VPN-1/FireWall-1 generates an Inspection Script, written in INSPECT. Inspection Code is compiled from the script and loaded to the VPN/Firewall Modules on the network's FireWalled enforcement points.

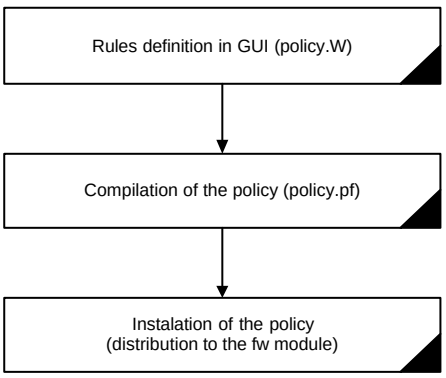
> Inspection script is an ASCII file (`*.pf`) in the INSPECT language. This file can be generated from a Security Policy (`*.W` file), or is written using a text editor. The file may be any combination of generated and manually written code. Its normal place to be is in the /conf FW1 directory named according to the name given in the policy editor.

The INSPECT Compiler (fwc command) is a single phase compiler. It translates the Inspection Script in two stages:

- During the preprocessor stage, any C preprocessor directives are carried out.
- In the second stage, the INSPECT Script is transformed into low level Inspection Code.

> Inspection code is file (`*.fc`) compiled from an Inspection Script (`*.pf`). Its position is also at /conf directory of FW1 from which it may be fetched by other firewall modules. It can be loaded by fw load command.

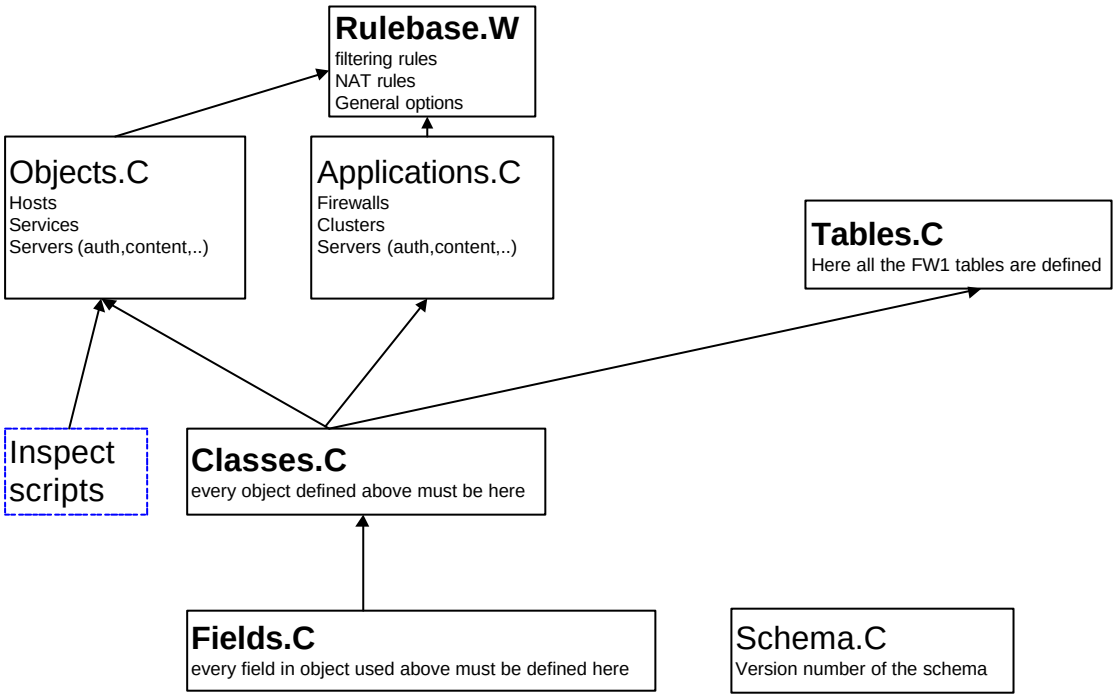
The Inspection Code is transmitted on a secured control channel from the Management Station — the computer on which the Security Policy was defined to the VPN-1/FireWall-1 daemons on the network objects that enforce the policy.
The VPN-1/FireWall-1 daemons load the Inspection Code into the VPN/FireWall Modules. The Inspection Code is run on a stack-based virtual machine. The virtual machine is placed in the FireWalled machine's kernel and inspects every IP packet passing through the machine.



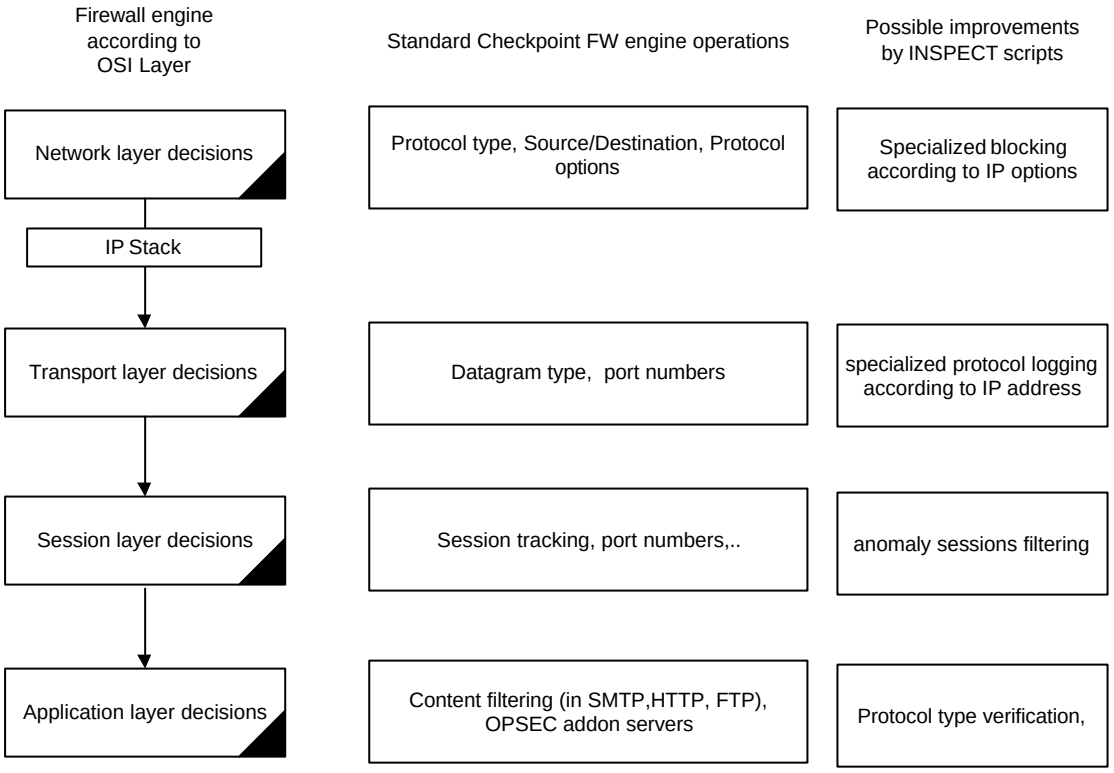
Administrators can tailor Inspection Scripts to their specialized security requirements in one of three ways:

1. By creating a User-Defined Service object using the GUI. For more information, see Chapter 6, “Services,” in *VPN-1/FireWall-1 Administration Guide.*

2. By including INSPECT scripts in `$FWDIR/lib/user.def` using the C preprocessor `#include` directive.
3. By creating an INSPECT Script named `$FWDIR/conf/defaultfilter` to be used as the default security policy (used before policy is loaded or cannot be loaded).



Here is a structure of a typical definition schema and place where inspect scripts may be visible. Inspect scripts are defined as services in Objects.C file. It may be there defined as handler() or possibly es exp().



Some overview of the functionality of CP FW1 engine and how INSPECT scripts can make it better.

Firewall connection functions

This chapter describes connection checking and tracking procedures and variables. It should be the primary source of knowledge for defining and verifying types of connections and doing appropriate actions with them.

```
 the partition of the cr_type 32 bit field :

 01234567    01234567    0123  4567    01234567
 >      <    >      <    >  <  >  <    >      <
  flags     encryption   ctrl  match     type
               args      bits  values   values
```

Connection types

CONN_TCP	(1)
CONN_UDP	(2)
CONN_ENC_A	(3)
CONN_ENC_B	(4)
CONN_IPSEC	(0x80)

CTRL_SRC_FIN	(1<<0)	bit mask !
CTRL_DST_FIN	(1<<1)	bit mask !
CTRL_TCP_ESTABLISHED	(1<<2)	bit mask !
CTRL_UDP_ESTABLISHED	(1<<2)	bit mask !

Matching cases

MATCH_BY_PROTOCOL	(0)
MATCH_BY_OFFSET	(1)
MATCH_BY_RPC	(2)
MATCH_BY_GETPORT	(3)
MATCH_BY_CALLIT	(4)
MATCH_BY_SEQACK_CHG	(5)

State table modifications

MAKE_ENTRY(type,ctrl,match,arg)	((type) | ((ctrl) << 12) | ((match) << 8) | ((arg) << 16))
ENTRY_TYPE(entry)	((entry) & 0xff)
ENTRY_TYPE_OLDCONN(entry)	((entry) & 0x7f)
ENTRY_CTRL(entry)	(((entry) >> 12) & 0xf)
ENTRY_MATCH(entry)	(((entry) >> 8) & 0xf)
ENTRY_ARG(entry)	(((entry) >> 16) & 0xff)
ENTRY_FLAGS(entry)	((entry) >> 24)
CHANGE_TYPE(entry,type)	(((entry) & 0xffffff00) | (type))
CHANGE_CTRL(entry,ctrl)	(((entry) & 0xffff0fff) | ((ctrl) << 12))
CHANGE_MATCH(entry,match)	(((entry) & 0xfffff0ff) | ((match) << 8))
CHANGE_ARG(entry,arg)	(((entry) & 0xff00ffff) | ((arg) << 16))
CHANGE_FLAGS(entry,flags)	(((entry) & 0x00ffffff) | ((flags) << 24))

ADD_TYPE(entry,type)	((entry) | (type))
ADD_CTRL(entry,ctrl)	((entry) | ((ctrl) << 12))
ADD_MATCH(entry,match)	((entry) | ((match) << 8))
ADD_ARG(entry,arg)	((entry) | ((arg) << 16))
ADD_FLAGS(entry,flags)	((entry) | ((flags) << 24))
REMOVE_TYPE(entry,type)	((entry) & ~(type))
REMOVE_CTRL(entry,ctrl)	((entry) & ~((ctrl) << 12))
REMOVE_MATCH(entry,match)	((entry) & ~((match) << 8))
REMOVE_ARG(entry,arg)	((entry) & ~((arg) << 16))
REMOVE_FLAGS(entry,flags)	((entry) & ~((flags) << 24))
_TCP_ESTABLISHED	MAKE_ENTRY(0, CTRL_TCP_ESTABLISHED, 0, 0)
_UDP_ESTABLISHED	MAKE_ENTRY(0, CTRL_UDP_ESTABLISHED, 0, 0)
_SRC_FIN	MAKE_ENTRY(0, CTRL_SRC_FIN, 0, 0)
_DST_FIN	MAKE_ENTRY(0, CTRL_DST_FIN, 0, 0)
_BOTH_FIN	(_SRC_FIN | _DST_FIN)

```
TCP_START_TIMEOUT standard (if not defined otherwise):
TCP_START_TIMEOUT 60

TCP_END_TIMEOUT standard (if not defined otherwise):
TCP_END_TIMEOUT   50
```

Auditing

Seems like this one is about tracking the sessions

```
 the partition of the r_cflags 32 bit field :
 01234567   01234567    01234567    01234567
  >     <   >      <    >      <    >      <
     anti spoofing      record src     flags
    conn        rconn
```

NO_CONN_ONEWAY	(0x00)
CONN_ONEWAY_A	(0x01)
CONN_ONEWAY_B	(0x02)
CONN_ONEWAY_EITHER	(0x03)
IS_ACCEPTED_A	(0x04)
IS_ACCEPTED_B	(0x08)
IS_ACCEPTED_EITHER	(0x0c)
MORE_INSPECTION	(0x10)
IS_NOT_TRACKED	(0)
IS_TRACKED_IN	(0x20)
IS_TRACKED_OUT	(0x40)
IS_TRACKED_UNK	(0x60)
IS_TRACKED_TRANS_IN	(0xa0)
IS_TRACKED_TRANS_OUT	(0xc0)
IS_TRACKED_TRANS_UNK	(0xe0)
TRACK_UNKNOWN	(0x80)
TRACK_DIRECTION_UNKNOWN	(0x60)
TRACK_HOW_NORMAL TRACK_HOW_TRANS	(0) (0x80)
SPOOF_CACHE_EMPTY ENTRY_ONEWAY(entry)	(0xffff0000) ((entry) & 0x03)

ENTRY_ACCEPTED(entry)	((entry) & 0x0c)
ENTRY_MORE_INSPECT(entry)	((entry) & 0x10)
ENTRY_TRACKED(entry)	((entry) & 0xe0)
ENTRY_SPOOF_CACHE(entry)	((entry) & 0xffff0000)
ENTRY_RSPOOF_CACHE(entry)	((((entry) & 0xff000000) >> 8) | (((entry & 0x00ff0000) << 8)))
ENTRY_SPOOF_CACHE_A(entry)	(((entry) >> 24) & 0xff)
ENTRY_SPOOF_CACHE_B(entry)	(((entry) >> 16) & 0xff)
CHANGE_ONEWAY(entry,way)	(((entry) & 0xfffffffc) | (way))
CHANGE_ACCEPTED(entry,acpt)	(((entry) & 0xfffffff3) | (acpt))
CHANGE_TRACKED(entry,track)	(((entry) & 0xffffff1f) | (track))
CHANGE_TRACK_DIR(entry,trk)	(((entry) & 0xffffff9f) | (trk << 5))
CHANGE_SPOOF_CACHE_A(entry,if_id)	(((entry) & 0x00ffffff) | ((if_id) << 24))
CHANGE_SPOOF_CACHE_B(entry,if_id)	(((entry) & 0xff00ffff) | ((if_id) << 16))
CHANGE_REC_FROM(entry,from)	(((entry) & 0xffff00ff) | ((from) << 8))
RECORD_SRC(src) TRACKED_TRANS(entry)	((src & 0x000000ff) << 8) ((((((entry) >> 5) & 1)|(((entry) >> 6) & 1)) * 7) << 5)
TRACK_DIRECTION(entry)	((entry) & 0x60)
TRACK_DIR_SHIFTED(entry)	(((entry) >> 5) & 1)
TRACK_HOW(entry)	((entry) & 0x80)

Connection logging

```
the partition of the r_pflags 32 bit field :
01234567    01234567    0123  4567  01234567
 >      <   >      <    >  <  >  <  >      <
                        action log     flags
```

SPOOF_CACHE_INVALID	(0x01)
TCP_FAST_MODE	(0x02)
CONN_ONEWAY_VIOLATE	(0x04)
PREMATCH_CRYPT	(0x08)
P_MORE_INSPECTION	(0x10)
ACTION_ACCEPT	(1)
ACTION_DROP	(2)
ACTION_VANISH	(3)
ACTION_REJECT	(4)
LOG_IPOPTIONS	(1)
LOG_CONN_ONEWAY	(2)
LOG_LOCAL_SPOOF	(3)
LOG_INV_TCP_SIZE	(4)
LOG_INV_UDP_SIZE	(5)
LOG_INV_ICMP_SIZE	(6)
LOG_INV_STREAM	(7)
ENTRY_SPOOF_CACHE_INVALID(entry)	((entry) & 0x00000001)
ENTRY_TCP_FAST_MODE(entry)	((entry) & 0x00000002)
ENTRY_CONN_ONEWAY_VIOLATE(entry)	((entry) & 0x00000004)
ENTRY_PREMATCH_CRYPT(entry)	((entry) & 0x00000008)
ENTRY_ACTION(entry)	(((entry) >> 12) & 0xf)
ENTRY_LOG(entry)	(((entry) >> 8) & 0xf)

ADD_ACTION(entry,action)	((entry) | ((action) << 12))
ADD_LOG(entry,log)	((entry) | ((log) << 8))
CHANGE_SPOOF_CACHE_INVALID(entry ,spc)	(((entry) & 0xfffffffe) | spc)

C_SRC	(0)
C_SPORT	(1)
C_DST	(2)
C_DPORT	(3)
C_IPP	(4)
C_LAST	C_IPP
X_HIDE_PORT	(0x80000000)
X_DST_STATIC	(0x40000000)
X_TCP_FIN	(0x20000000)
X_TCP_EST	(0x10000000)
X_REVERSE_UDP	(0x08000000)
X_FOLD	(0x04000000)
X_AUTH	(0x02000000)
X_SRC_DYNAMIC	(0x01000000)
X_DST_DYNAMIC	(0x00800000)
X_FLAG_MASK	(0xffff0000)
NOACCT	(0)
ACCT	(1)

X_TCPSEQ	(4)
MAKE_ENTRY(type,ctrl,match,arg)	((type) | ((ctrl) << 12) | ((match) << 8) | ((arg) << 16))
MAKE_PVAL(acode,pid)	((acode) | ((pid) << 4))
PVAL_ACT(pval)	((pval) & 0xf)
PVAL_ID(pval)	((pval) >> 4)
EX_INFINITE	0x7fffffff

SecuRemote security modes bit mask

USERC_OUTBOUND_VPN	0x00000001
USERC_OUTBOUND	0x00000010
USERC_INBOUND_VPN_PRIV	0x00000100
USERC_INBOUND_VPN	0x00000200
USERC_INBOUND_PRIV	0x00001000
USERC_INBOUND	0x00002000
USERC_FORWARD	0x00004000
USERC_ENABLE_LOGS	0x00020000

Have to define “SECUREMOTE” to use these:

USERC_IGNORE_VERIFIED USERC_RECORD_VERIFIED	0x0000 0x0001
USERC_BLOCK_VERIFIED USERC_IN_ENC_DOMAIN_CLEAR	0x0002 0x0001

URL logging

These values are shared between the fwurlog module and Inspect. It is written in inspect (fwauth.def) to the url_log_conns table in the following format:

<conn; tcp_seq, flags, rule_no, ufp_mask>

tcp_seq - sequence # at which streaming will be started (SYN+1)
flags :

- **LOG_ALL** (all URLs in connection should be logged)
- **LOG_FIRST** (only first URL should be logged)
- **SECURE_LOG_FIRST** (same as LOG_FIRST but streaming is continued for additional security)
- **WITH_AUTH** (indicated that the connection was with client authentication and therefore username should also be logged)
- **USE_ACTIVE_STREAMING** (use active TCP streaming (the default is passive))

Internal use only:

- **NO_UFP** (used when URL logging is done without the UFP cache feature)

URLOG_CONNS_NVALS	4
URLOG_LOG_METHOD_MASK	3 (2 bits mask)
URLOG_NO_LOG	0
URLOG_LOG_ALL	1
URLOG_LOG_FIRST	2
URLOG_LOG_FIRST_SECURE	3
URLOG_WITH_AUTH	4
URLOG_STREAM_INITIALIZED	8
URLOG_LOG_DONE	16
URLOG_NO_UFP	32
URLOG_USE_ACTIVE_STREAMING	64

FW1 Tables functions

This chapter describes functions and variables that represent or do something with FW1 tables.

TIMEOUT Definitions

Here come all the defaults that are not defined elsewhere (like policies or inspect scripts)

TCP_TIMEOUT	3600
ENCAP_TIMEOUT	4000
TCP_START_TIMEOUT	60
TCP_EXTENDED_TIMEOUT	TCP_TIMEOUT
PENDING_TIMEOUT	60
LOG_TIMEOUT	61
UDP_TIMEOUT	30
TRAP_TIMEOUT	10
AUTH_TIMEOUT	60
ENCDOMAIN_TIMEOUT	5
AU_PORT_TIMEOUT	1800
VIOLATED_TIMEOUT	180

```
/**************************************************
*    The encryption pending table timeout        *
*    Should be greater than RDP cypher protocol   *
*    timeout RDP_MAX_TTL defined in rdp c source  *
**************************************************/
```

ENCRYPTION_PENDING_TIME	180
ENCRYPTION_PENDING_SHORT_TIME	90
DECRYPTION_PENDING_TIME	120
RDP_TABLE_TIME	60
USERC_TIMEOUT	900
USERC_BIND_TIMEOUT	3600

Defines for logical servers table *

In case there is no definition of “LOGICAL_SERVERS_TABLE” these are defined:

LOGICAL_SERVERS_TABLE	logical_servers_table
LOGICAL_SERVERS_LIST_TABLE	logical_servers_list_table
LOGICAL_CACHE_TABLE	logical_cache_table
SERVER_ALIVE_TIMEOUT	60
LOGICAL_CACHE_TIMEOUT	1800

LOGICAL_CACHE_SIZE	1000

DCE RPC

dcerpc_maps	dynamic sync refresh expires 86400 keep;
dcerpc_binds	dynamic sync refresh expires TCP_TIMEOUT;
dcerpc_sessions	dynamic sync refresh expires TCP_TIMEOUT;
dcerpc_epm_requests	dynamic sync expires 20;

DCOM

dcom_objects	dynamic sync refresh expires 86400 keep;
dcom_remote_activations	dynamic sync refresh expires 60;

USER CLIENT ENCRYPTION

<src,0;0,kbuf = user name> or <src,rulenum;0,0> (ignore user database) or <src,rulenum;1,0> (intersect dst with user database)

userc_rules	dynamic expires USERC_TIMEOUT sync kbuf 2;

<ip; gwip>, where ip is a SecuRemote client, and the value is the address we will use for encapsulation to it

userc_encapsulating_clients	dynamic refresh sync keep expires ENCAP_TIMEOUT;

<dst;0> (never trap again) or <dst;1> (trap again only if rule ignores user database)

userc_dont_trap	dynamic expires 10;

<client ip;kbuf> , where kbuf holds the hash of the public key of the client

userc_bind	dynamic expires USERC_BIND_TIMEOUT sync keep kbuf 1;

<client ip;kbuf> , where kbuf holds a userc_entry_t structure

userc_key	dynamic expires USERC_TIMEOUT sync keep kbuf 1;

<client_ip;kbuf(username)>

userc_slan	dynamic keep kbuf 1;

<client_ip>

userc_dtm_cache	dynamic expires 30;

If "SECUREMOTE" is defined:

userc_noncrypt_ports	dynamic;
userc_verified_connections	dynamic;
userc_blocked_connections	dynamic;

Suspicious Action Monitoring

SAM restricted IP's table

entry format: [<ip>;<ip-flag>,<log-flag>,<notify>]

where: ip-flag is one of:

- SAM_RESTR_IP
- SAM_RESTR_SRC
- SAM_RESTR_DST
- SAM_RESTR_CONN_SRC
- SAM_RESTR_CONN_DST

log-flag is one of: 1,2,3,4

sam_blocked_ips = dynamic sync keep;

SAM restricted connections table

entry format: [<src-ip>,<dst-ip>,<dport>,<ip_p>;<log-flag>,<notify>]

sam_blocked_servs	dynamic sync keep;

Global Table of Session *

Connections table format:

<src,sport,dst,dport,ip_p;kbuf,type,flags>

old_connections	dynamic sync expires TCP_TIMEOUT keep kbuf 2;
proxied_conns	dynamic keep;
ftp_restrictions	dynamic;
connections	dynamic refresh sync expires TCP_START_TIMEOUT expcall KFUNC_CONN_EXPIRE kbuf 1

If "SECUREMOTE" is defined/used there are also these options in connection table:

	implies userc_verified_connections

If not:

	implies ftp_restrictions

This comes to the end of connections table definition:

	hashsize 32768 limit 25000;
udp_enc_cln_table	dynamic expires USERC_TIMEOUT sync keep;

Tracked table format:

<src,sport,dst,dport,ip_p; start_time,num_packets, num_bytes,rule,num_updates, interface,last_seen>

or <0,src,sport,dst,dport,ip_p;conn> for translation entry

tracked	dynamic refresh expires 10000 expcall

	KFUNC_END_ACCT_CONN;

Pending Table

This tables should be used for storing partial connection entries, for various services. The format and actual timeout are service dependent, but should resemble the connections format as much as possible.

pending	dynamic refresh sync expires PENDING_TIMEOUT kbuf 1;

Log and Trap tables

if LOG_TIMEOUT > 0

logged	dynamic expires LOG_TIMEOUT;

MAKE_ALERT(alert_tab, alert) alert_tab = format alert { };

trapped tcp_timeouts	dynamic expires TRAP_TIMEOUT; dynamic;

Remote Procedure Call (RPC)

rcp_serv_hosts has <host> if host rpc mappings are loaded and <host,ip_p,port,rpcnum> for every rpc port on host

rpc_serv_hosts	dynamic sync expires 700;
rpc_serv	dynamic refresh sync expires 800;
rpc_sessions	dynamic refresh sync expires UDP_TIMEOUT;

pmap_req is a temporary table used while trying to add entries to rpc_serv according to packets going thru the INSPECT machine.

pmap_req	dynamic sync expires 10;

could not get rpc mapping info from each <host> here

pmap_not_responding	dynamic sync expires 120;

Ping Server

<ip,magic;last_time,time_to_die,recheck_time> */

check_alive	dynamic;

AUTHENTICATION

auth_services	dynamic;
client_auth	dynamic sync expires AUTH_TIMEOUT kbuf 3;
client_auth_username	dynamic expires 60 kbuf 1;
client_was_auth	dynamic refresh expires AU_PORT_TIMEOUT;
autoclntauth_fold	dynamic expires 60;
session_requests	dynamic expires 180;
sso_requests	dynamic expires 180;
session_auth	dynamic sync expires AUTH_TIMEOUT;

ENCRYPTION

encryption_requests	dynamic expires ENCRYPTION_PENDING_SHORT_TIME;
rejected_encryptions	dynamic expires ENCRYPTION_PENDING_TIME;
decryption_pending rdp_table	dynamic expires DECRYPTION_PENDING_TIME kbuf 1; dynamic expires RDP_TABLE_TIME;
crypt_more_inspection	dynamic expires 60;
crypt_resolver_db	dynamic expires 60;
crypt_resolver_uptag	dynamic keep sync;
first_packet_was_decrypted	dynamic expires 720;

BALANCING (LOGICAL SERVERS)

if NLOGIC > 0

logical_requests	dynamic sync expires ENCRYPTION_PENDING_TIME;
LOGICAL_SERVERS_TABLE	dynamic;
LOGICAL_SERVERS_LIST_TABLE	dynamic;
LOGICAL_CACHE_TABLE	dynamic sync refresh expires LOGICAL_CACHE_TIMEOUT limit LOGICAL_CACHE_SIZE;

STATEFUL ICMP

This option can be defined in global options

icmp_connections	dynamic sync refresh expires TCP_START_TIMEOUT;
icmp_requests	{ ICMP_ECHO, ICMP_TSTAMP, ICMP_IREQ, ICMP_MASKREQ };
icmp_replies	{ ICMP_ECHOREPLY,ICMP_TSTAMPREPLY, ICMP_IREQREPLY, ICMP_MASKREPLY};
icmp_errors	{ ICMP_UNREACH, ICMP_SOURCEQUENCH, ICMP_TIMXCEED, ICMP_PARAMPROB, ICMP_REDIRECT };

Embedded license violation

forbidden_tab	dynamic expires 300;

IIOP

iiop_port_tab	dynamic;
iiop_servers	dynamic refresh expires 600;
iiop_requests	dynamic refresh expires 30;

ip ufp cache

ipufp_cache	dynamic expires 1200 limit 25000;
urlog_conns	dynamic refresh expires TCP_TIMEOUT limit 25000;

stateful DNS

dns_requests	dynamic sync refresh expires 30;

CONN_ONE_WAY violation

bad_connections	dynamic refresh sync expires VIOLATED_TIMEOUT hashsize 32768 limit 25000;

Kernel variables for tables

Here some kernel variables are defined that are used in different functions in FW1.

IDs of special kernel tables

FWMAXTABS=8192

if defined(NOKIA_ENABLE_FLOW)

NOKFW_FLOWS_TABLE_ID	8191

FWDOM_TABLE_ID	8190
FWX_FORW_TABLE_ID	8189
FWX_BACKW_TABLE_ID	8188
FWX_ALLOC_TABLE_ID	8187
FWX_ARP_TABLE_ID	8186
FWLIC_TABLE_ID	8185
FWFRAG_TABLE_ID	8184
FWHOLD_TABLE_ID	8183
FWESTAB_TABLE_ID	8182
FWLIC_USR_TABLE_ID	8181
FWSKIP_TABLE_ID	8180
FWSKIP_CONN_TABLE_ID	8179
FWSKIP_REQ_TABLE_ID	8178
FWHOST_IP_ADDRS	8177
FWMANUAL_TABLE_ID	8176
FWX_FRAG_TABLE_ID	8175
FWX_AUTH_TABLE_ID	8174
FWSYNATK_TABLE_ID	8173
FWH323_TABLE_ID	172
FWSKIP_KEYID_ID	8171
FW_INBOUND_SPI_TABLE_ID	8170
FW_ISAKMP_ESP_TABLE_ID	8169
FW_SA_REQUESTS_TABLE_ID	8168
FW_ISAKMP_AH_TABLE_ID	8167
FW_CRYPTLOG_TABLE_ID	8166
FW_IPSEC_USERC_DONT_TRAP_TABLE_ID	8165
FWBRIDGE_TABLE_ID	8164
FW_IPSEC_SHORTCUT_TABLE_ID FW_OUTBOUND_SPI_TABLE_ID	8163 8162
FW_IKE_SA_TABLE_ID	8161
FWBB_RT_MAC_TABLE_ID	8160
FWBB_RT_IP_TABLE_ID	8159
FWBB_ARP_REQ_TAB_ID FWX_DYN_CONN_TABLE_ID	8158 8157

FWX_CTL_DYN_TABLE_ID	8156
FWX_IP_LOOKUP_TABLE_ID	8155
FW_SAVED_KBUF_TABLE_ID	8154

ifdef MULTI_MOD

FWMM_PACKET_TABLE_ID	153
FWSM_IOCTL_TABLE_ID	8152
FW_ROUTE_TABLE_ID	8151

FW_TCP_STREAMS_TABLE_ID	8150
FW_PACKET_HASH_TABLE_ID	149

if !defined(SECUREMOTE)

AUTO_REFRESH_TABLE_ID	8148

FW_IPSEC_MTU_TABLE_ID	8147

Limit INSPECT tables to FWMAXTABS-192

NUM_KERN_TABS	192

Traps

Daemon traps

UTH_INVOKE	0x101
AUTH_INTRCPT	0x102
UTH_MATCH	0x103
AUTH_KCONN_INVOKE	0x104
URLOG_LOG_URL	0x105
MAP_FETCH	0x110
ENCRYPT_INVOKE	x120
ENCRYPT_INTRCPT	0x121

User Encryption (GW side) old (V2.1) trap:

At this stage we need only check that the destination is in the encryption domain of the gateway, and if it is, add this connection to the connections table (with its key). The daemon is also responsible for sending the log. After completing the check, the packet, which caused the invoke, will be released (regardless of the result of the check).

USERC_SERVER_INVOKE	x122

User Encryption (GW side) new (V3.0) trap:

At this stage we need only check that the destination is in the encryption domain of the gateway, and if it is, add this connection to the connections table (with its key). The daemon is also responsible for sending the log. After completing the check, the packet, which caused the invoke, will be released (regardless of the result of the check).

USERC_NEW_SERVER_INVOKE	x128
USERC_NEW_NEW_SERVER_INVOKE	x133

User Encryption (PC side):

The daemon running on the PC is requested to establish a session with a GW. The session key will be added to the session table. The packet which caused the invoke will be relased.

USERC_ENCRYPT_INVOKE	x123

The daemon running on SecuRemote Gold is asked to establish a connection in connection table using knowkn or negotiated SA.

USERC_ENCRYPT_INVOKE_GOLD	0x12a
USERC_RESOLVE	0x12d
USERC_LOG	0x12e

Invoked when encountering a SKIP packet for which there is no established secret key between the source and destination.

SKIP_DECRYPT	0x124

Invoked when encountering an IPSec packet without key a management protocol for the first time

MANUAL_DECRYPT	0x125
SPI_REQUEST	0x135
SPI_NOTIFY	0x129

Invoked when a SecuRemote ISAKMP client is initiating a new conenction and has no entry in userc_rules.

UPDATE_ISAKMP_USER	0x13b

The Decrypt-Invoke trap moves an entry from the decryption pending table to the connections table, given that the encryption parameters of the key of that entry match those of the rule.

DECRYPT_INVOKE	0x11f
BALANCE_INVOKE	0x130
BALANCE_DOMAIN	0x131

Kernel logging mechanism

CRYPT_LOG	0x12b
CLIENT_AUTH_LOG	0x137

SecureIP

CLAUTH_LOG	0x300
CLAUTH_INVOKE	0x301

Kernel logging mechanism for logging encrypted connections.

CRYPT_LOG_CONN	x12c

CRYPT_LOG_POOL	0x12f

User Encryption (PC side):

The daemon running on the PC is requested to RDP several GWs and establish a session with the first that answers.

ENCRYPT_INVOKE_MUL	0x131
EBUG_TUNNEL	0x166
HOST_REPORT	0x170
SYNATK_REPORT	0x180
TRAP_EXECUTE	x190
ACCT_REPORT	x200
ADD_CONN	x210
DEL_CONN	x211
CONN_UPDATE	x213
SYNC_INIT	0x214

Reason codes

RCODE_TCP_SERV	1
RCODE_UDP_SERV	2
RCODE_SMALL_PORT	3
RCODE_TCP_FASTMODE_PORT	4
RCODE_WRONG_HOST	5
RCODE_LOCAL_SPOOF	6
RCODE_CONN_ONEWAY	7
RCODE_INVALID_TCP_SIZE	8
RCODE_INVALID_UDP_SIZE	9
RCODE_LAND_ATTACK	10
RCODE_INVALID_ICMP_SIZE	11
RCODE_TCP_EST	12
RCODE_STREAM_VIOLATION	13

Kernel traps

KFUNC_FTPPORT	0
KFUNC_GETNET	1
KFUNC_ENCRYPT	2
KFUNC_DECRYPT	3
KFUNC_CONN_EXPIRE	4
KFUNC_KBUF_DUP	5
KFUNC_MASK_GET	6
KFUNC_TABLES	7
KFUNC_IS_MY_IPADDR	8
KFUNC_USERC_EXPIRE	9
KFUNC_KBUF_SAFE_DUP	10

KFUNC_TCP_ESTABLISHED	11
FUNC_XLATE_FORW	12
FUNC_XLATE_BACKW	13
FUNC_XLATE_FOLD	14
KFUNC_XLATE_ANTICIPATE	15
KFUNC_END_ACCT_CONN	16
KFUNC_SQLNET	17
KFUNC_INIT_H225	18
KFUNC_FOLLOW_H225	19
KFUNC_INIT_H245	20
KFUNC_FOLLOW_H245	21
KFUNC_GET_RTP	22
KFUNC_NOT_IPSEC_ENCAPS	23
FUNC_IS_MUT_GW	24
KFUNC_GET_GW	25
KFUNC_LOAD_BALANCE	26
KFUNC_LOG_ENCRYPTION_CONN	27
KFUNC_XLATE_PREDICT	28
KFUNC_IPPOOL_ALLOCATE	29
FUNC_IS_IN_GW	30
KFUNC_SET_XLATED_IF	31
KFUNC_FIND_STR	32
KFUNC_IS_GWCLUSTER	33
KFUNC_DNS_FILTER	34
KFUNC_FOLLOW_H323	35
KFUNC_H323_SAME_GW	36

TCP/IP variables

In order to make a script more readable it is easier to use predefined values of certain fields.

ip_tos	[1 : 1]
ip_len	[2 : 2, b]
ip_id	[4 : 2, b]
ip_off	[6 : 2, b]
ip_ttl	[8 : 1]
ip_p	[9 : 1]
ip_sum	[10 : 2, b]
ip_src	[12 , b]
ip_dst	[16 , b]
th_sport	[20 : 2, b]
th_dport	[22 : 2, b]
th_seq	[24 , b]
th_ack	[28 , b]
th_flags	[33 : 1]
th_win	[34 : 2, b]
th_sum	[36 : 2, b]
th_urp	[38 : 2, b]
TH_FIN	0x1
TH_SYN	0x2
TH_RST	0x4
TH_PUSH	0x8
TH_ACK	0x10
TH_URG	0x20
uh_sport	[20 : 2, b]
uh_dport	[22 : 2, b]
uh_ulen	[24 : 2, b]
uh_sum	[26 : 2, b]
icmp_type	[20 : 1]
icmp_code	[21 : 1]
icmp_cksum	[22 : 2, b]
icmp_id	[24 : 2, b]
icmp_seq	[26 : 2, b]
icmp_idseq	[24 , b]
icmp_ip_tos icmp_ip_len	[29 : 1] [30 : 2, b]
icmp_ip_id	[32 : 2, b]
icmp_ip_off	[34 : 2, b]
icmp_ip_ttl	[36 : 1]
icmp_ip_p	[37 : 1]
icmp_ip_sum	[38 : 2, b]

icmp_ip_src	[40 , b]
icmp_ip_dst	[44 , b]
icmp_th_sport	[48 : 2, b]
icmp_th_dport	[50 : 2, b]
icmp_uh_sport	[48 : 2, b]
icmp_uh_dport	[50 : 2, b]
icmp_icmp_id	[52 : 2, b]
icmp_icmp_seq	[54 : 2, b]
ICMP_ECHOREPLY	0x0
ICMP_UNREACH	0x3
ICMP_SOURCEQUENCH	0x4
ICMP_REDIRECT	0x5
ICMP_ECHO	0x8
ICMP_TIMXCEED	0xb
ICMP_PARAMPROB	0xc
ICMP_TSTAMP	0xd
ICMP_TSTAMPREPLY	0xe
ICMP_IREQ	0xf
ICMP_IREQREPLY	0x10
ICMP_MASKREQ	0x11
ICMP_MASKREPLY	0x12
rm_xid	[28 , b]
rm_direction	[32 , b]
cb_rpcvers	[36 , b]
cb_prog	[40 , b]
cb_vers	[44 , b]
cb_proc	[48 , b]
cb_cred_oa_flavor	[52 , b]
cb_cred_oa_base cb_cred_oa_length	[56 , b] [60 , b]
cb_verf_oa_flavor	[64 , b]
cb_verf_oa_base	[68 , b]
cb_verf_oa_length	[72 , b]
pm_prog	[76 , b]
pm_vers	[80 , b]
pm_prot	[84 , b]
pm_port	[88 , b]
rdp_magic	[28 , b]
rdp_cmd	[32 , b]
cr_src	[44 , b]
cr_dst	[48 , b]
cr_sport	[52 : 2, b]
cr_dport	[54 : 2, b]
cr_p	[56 : 1]
sr_cr_src	[36 , b]

sr_cr_dst	[40 , b]
rip_cmd	[28 : 1]
rip_vers	[29 : 1]
RIP_PORT	0x208
RIPCMD_REQUEST	0x1
RIPCMD_RESPONSE	0x2

OSPF

It was never tested by checkpoint. Use with caution.

OSPF_IPPROTO	0x59
ospf_version	[20 : 1]
ospf_type	[21 : 1]
ospf_length	[22 : 2, b]
ospf_rtr_id	[24 , b]
ospf_area_id	[28 , b]
OSPF_MON	0x0
OSPF_HELLO	0x1
OSPF_DB_DESCRIPT	0x2
OSPF_LSR	0x3
OSPF_LSU	0x4
OSPF_ACK	0x5

BGP

It was never tested by checkpoint. Use with caution.

BGP_PORT	0xb3
bgp_marker	[40 : 16]
bgp_length	[56 : 2, b]
bgp_type	[58 : 1]
BGP_OPEN	0x1
BGP_UPDATE	0x2
BGP_NOTIFICATION	0x3
BGP_KEEPALIVE	0x4

EGP

It was never tested by checkpoint. Use with caution.

EGP_IPPROTO	0x8
egp_ver	[20 : 1]
egp_type	[21 : 1]
egp_code	[22 : 1]
egp_status	[23 : 1]
egp_chksum	[24 : 2, b]
egp_system	[26 : 2, b]

egp_seqnum	[28 : 2, b]
EGP_PKT_NR	0x1
EGP_PKT_POLL	0x2
EGP_PKT_ACQUIRE	0x3
EGP_PKT_HELLO	0x5
EGP_PKT_ERROR	0x8
udp_header_sz	28

tcp states

syn	{ th_flags & TH_SYN };
fin	{ th_flags & TH_FIN };
rst	{ th_flags & TH_RST };
ack	{ th_flags & TH_ACK };
first	{ th_flags & TH_SYN, not (th_flags & TH_ACK) };
established	{ (th_flags & TH_ACK) or ((th_flags & TH_SYN) = 0) };
not_first	{ not (th_flags & TH_SYN) };
last	{ th_flags & TH_FIN, th_flags & TH_ACK };
tcpdone	{ fin or rst };

Log Formats definition

```
//LOG_FORMAT and ALERT_COMMAND SYNTAX
// format_name = format <"![leading_string] command_line_to_execute"> {
//      <"string1", format_type1, data_field1>,
//      <"string2", format_type2, data_field2> ...
// };
// data fields in packet:       read "/etc/fw/lib/tcpip.def" and equivalents
// format types available:      ipaddr service port proto int uint hex
// entire section is optional:  <"![leading_string] command_line_to_execute">
//

short = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>
};

auth_short = format {
        <"src", ipaddr, src>,
        <"user", string, 0>,
        <"reason", string, 0>,
        <"service", string, dport>
};

web_short = format {
        <"src", ipaddr, src>,
        <"user", string, 0>,
        <"reason", string, 0>,
        <"service", string, dport>,
        <"dstname", string, 0>
};

icmp_short = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>
};

rpc_short = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"rpc_prog", int, cb_prog>
```

```
};

long = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"len", int, packetlen>,
        <"rule", rule, sr1>
#ifdef FWXLATE_ACTIVE
        ,<"xlatesrc", ipaddr, xlatesrc>,
        <"xlatedst", ipaddr, xlatedst>,
        <"xlatesport", port, xlatesport>,
        <"xlatedport", port, xlatedport>
#endif
};

synatk = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"rule", rule, 0>,
        <"message", string, 0>
};

auth = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", string, dport>,
        <"s_port", port, sport>,
        <"user", string, 0>,
        <"rule", rule, sr1>,
        <"reason", string, 0>,
        <"device", string, 0>,
        <"file", string, 0>
};

#ifdef FWEMBEDED
sessauth = format {
    <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
    <"dst", ipaddr, dst>,
```

```
    <"service", port, dport>,
    <"s_port", port, sport>,
    <"user", string, 0>,
    <"rule", rule, sr1>,
    <"reason", string, 0>,
    <"file", string, 0>
};
#endif

web_long = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", string, dport>,
        <"s_port", port, sport>,
        <"user", string, 0>,
        <"rule", rule, sr1>,
        <"reason", string, 0>,
        <"res_action", string, 0>,
        <"resource", string, 0>,
        <"category", mask, 0>,
        <"cat_server", string, 0>,
        <"dstname", string, 0>
};

mail_long = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", string, dport>,
        <"s_port", port, sport>,
        <"agent", string, 0>,
        <"error notification:", string, 0>,
        <"orig_from", string, 0>,
        <"orig_to", string, 0>,
        <"from", string, 0>,
        <"to", string, 0>,
        <"rule", rule, sr1>,
        <"reason", string, 0>
};

icmp_long = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"rule", rule, sr1>,
```

```
        <"icmp-type", int, icmp_type>,
        <"icmp-code", int, icmp_code>
#ifdef FWXLATE_ACTIVE
        ,<"xlatesrc", ipaddr, xlatesrc>,
        <"xlatedst", ipaddr, xlatedst>
#endif
};

rpc_long = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"rpc_prog", int, cb_prog>,
        <"rule", rule, sr1>
};

account = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"rule", rule, sr1>,
        <"elapsed", time, 0>,
        <"start_time", timestmp, 0>,
        <"packets", uint, 0>,
        <"bytes", uint, 0>,
        <"user", string, 0>,
        <"res_action", string, 0>,
        <"reason", string, 0>,
        <"category", mask, 0>,
        <"cat_server", string, 0>,
        <"resource", string, 0>,
        <"from", string, 0>,
        <"to", string, 0>
#ifdef FWXLATE_ACTIVE
        ,<"xlatesrc", ipaddr, xlatesrc>,
        <"xlatedst", ipaddr, xlatedst>,
        <"xlatesport", port, xlatesport>,
        <"xlatedport", port, xlatedport>
#endif
};

live_conns_acct = format {
        <"proto", proto, ip_p>,
```

```
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"bytes", uint, 0>,
        <"packets:", uint, 0>,
        <"elapsed", time, 0>,
        <"command:", int, 0>
};

live_conns = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"command:", int, 0>
};

auth_crypt = format {
        <"src", ipaddr, src>,
        <"srckeyid", string, 0>,
        <"dstkeyid", string, 0>,
        <"user", string, 0>,
        <"rule", rule, sr1>,
        <"reason", string, 0>,
        <"scheme:", string, 0>,
        <"methods:", string, 0>
};

auth_verify = format {
        <"src", ipaddr, src>,
        <"user", string, 0>,
        <"reason", string, 0>
};

crypt = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"srckeyid", string, 0>,
        <"dstkeyid", string, 0>,
        <"rule", rule, sr1>,
        <"user", string, 0>,
```

```
        <"encryption failure:", string, 0>,
        <"decryption failure:", string, 0>,
        <"success reason:", string, 0>,
        <"scheme:", string, 0>,
        <"SPI:", string, 0>,
        <"methods:", string, 0>
#ifdef FWXLATE_ACTIVE
    ,<"xlatesrc", ipaddr, xlatesrc>,
    <"xlatedst", ipaddr, xlatedst>,
        <"xlatesport", port, xlatesport>,
        <"xlatedport", port, xlatedport>
#endif
};

crypt_icmp = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"icmp-type", int, icmp_type>,
        <"icmp-code", int, icmp_code>,
        <"srckeyid", string, 0>,
        <"dstkeyid", string, 0>,
        <"rule", rule, sr1>,
        <"user", string, 0>,
        <"encryption failure:", string, 0>,
        <"decryption failure:", string, 0>,
        <"success reason:", string, 0>,
        <"scheme:", string, 0>,
        <"methods:", string, 0>
#ifdef FWXLATE_ACTIVE
    ,<"xlatesrc", ipaddr, xlatesrc>,
    <"xlatedst", ipaddr, xlatedst>
#endif
};

crypt_rpc = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"rpc_prog", int, cb_prog>,
        <"srckeyid", string, 0>,
        <"dstkeyid", string, 0>,
        <"rule", rule, sr1>,
        <"user", string, 0>,
```

```
        <"encryption failure:", string, 0>,
        <"decryption failure:", string, 0>,
        <"success reason:", string, 0>,
        <"scheme:", string, 0>,
        <"methods:", string, 0>
#ifdef FWXLATE_ACTIVE
    ,<"xlatesrc", ipaddr, xlatesrc>,
    <"xlatedst", ipaddr, xlatedst>,
        <"xlatesport", port, xlatesport>,
        <"xlatedport", port, xlatedport>
#endif
};

keyinst = format {
        <"srckeyid", string, 0>,
        <"key update for", string, 0>,
        <"signed by", string, 0>
};

rpcauth =  format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", string, dport>,
        <"s_port", port, sport>,
        <"user", string, 0>,
        <"rule", rule, sr1>,
        <"rpc_prog", int, cb_prog>,
        <"reason", string, 0>
};
#define ip_hl [0:1]
#define IPHLEN ((ip_hl & 0xf) * 4)
badip_form = format {
        <"proto", proto, ip_p>,
        <"service", port, [(IPHLEN+2):2, b]>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"s_port", port, [(IPHLEN):2, b]>,
        <"h_len", int, IPHLEN>,
        <"ip_vers", int, ip_hl / 16>,
        <"rule", rule, sr1>
};
icmp_badip_form = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
```

```
        <"icmp-type", hex, [(IPHLEN):1]>,
        <"icmp-code", hex, [(IPHLEN+1):1]>,
        <"h_len", int, IPHLEN>,
        <"ip_vers", int, ip_hl / 16>,
        <"rule", rule, sr1>
};
host_count_format = format {
        <"license violation detected", ipaddr, src>
};
isakmp = format {
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"srckeyid", string, 0>,
        <"dstkeyid", string, 0>,
        <"IKE Log:", string, 0>,
        <"Negotiation Id:", string, 0>
};
bad_conn = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"rule", rule, sr1>,
        <"reason:", reasoncode, sr10>,
        <"host:", ipaddr, sr11>,
        <"port:", port, sr12>
};
sam = format {
        <"src", ipaddr, src>,
        <"s_port", port, sport>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"request", string, 0>,
        <"target", string, 0>,
        <"expire", string, 0>
};
clauth_short = format {
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"user", string, 0>
};
clauth_long = format {
        <"proto", proto, ip_p>,
```

```
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"rule", rule, sr1>,
        <"user", string, 0>,
        <"len", int, packetlen>,
        <"xlatesrc", ipaddr, xlatesrc>,
        <"xlatedst", ipaddr, xlatedst>,
        <"xlatesport", port, xlatesport>,
        <"xlatedport", port, xlatedport>,
        <"srcname", string, 0>
};
pool_event = format {
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"xlatesrc", ipaddr, xlatesrc>,
        <"xlatedst", ipaddr, xlatedst>,
        <"user", string, 0>,
        <"IP Pool:", string, 0>
};
#define rpc_badip_form badip_form
```

Initial definitions

Here the code would be executed after successful installation of a rulebase on the firewall module.

```
ENTRY_LOG(r_pflags) or ENTRY_ACTION(r_pflags),
      (set sr1 0, set sr11 0, set sr12 0,
            (ENTRY_LOG(r_pflags) = LOG_IPOPTIONS, IPOPTNS_LOG)
        or
            (ENTRY_LOG(r_pflags) = LOG_CONN_ONEWAY, set sr10
RCODE_CONN_ONEWAY,
             log bad_conn)
        or
            (ENTRY_LOG(r_pflags) = LOG_LOCAL_SPOOF, set sr10
RCODE_LOCAL_SPOOF,
             log bad_conn)
        or
            (ENTRY_LOG(r_pflags) = LOG_INV_TCP_SIZE,
             set sr10 RCODE_INVALID_TCP_SIZE,
             log bad_conn)
        or
            (ENTRY_LOG(r_pflags) = LOG_INV_UDP_SIZE,
             set sr10 RCODE_INVALID_UDP_SIZE,
             log bad_conn)
        or
            (ENTRY_LOG(r_pflags) = LOG_INV_ICMP_SIZE,
             set sr10 RCODE_INVALID_ICMP_SIZE,
             log bad_conn)
        or
            (ENTRY_LOG(r_pflags) = LOG_INV_STREAM,
             set sr10 RCODE_STREAM_VIOLATION,
             log bad_conn)
        or 1
      ),(
            (ENTRY_ACTION(r_pflags) = ACTION_ACCEPT, accept)
        or
            (ENTRY_ACTION(r_pflags) = ACTION_DROP, drop)
        or
            (ENTRY_ACTION(r_pflags) = ACTION_VANISH, vanish)
        or
            (ENTRY_ACTION(r_pflags) = ACTION_REJECT, reject)
      );

/*
 * Accept established TCP packets for selected services in TCP Fast Mode.
 */
```

```
#ifdef TCP_FASTMODE_ACTIVE
accept ENTRY_TCP_FAST_MODE(r_pflags);
#endif

/*
 * Land attack protection
 */

inbound all@all {
        vanish
                src=dst, sport=dport, tcp,
                set sr10 RCODE_LAND_ATTACK, set sr11 0, set sr12 0, set sr1 0,
                log bad_conn;
}

/*
 * Do not initiate new A/T entries if this is a reply packet of an already
 * known connection
 */

r_cdir = 2, set r_xlate -1;

set r_tab_status (call KFUNC_TABLES<>);

/*
 * Set timeouts for tcp services
 *
 * In order to change the tcp session timeout for specific services, add a line
 * "ADD_TCP_TIMEOUT(port,timeout)," before the line "ADD_TCP_TIMEOUT(0,0)"
 * where port is the TCP service port and timeout is the desired timeout.
 */

#ifndef FTP_CONTROL_TIMEOUT
#define FTP_CONTROL_TIMEOUT TCP_TIMEOUT
#endif

#define ADD_TCP_TIMEOUT(port,to) (record <port;to> in tcp_timeouts)

(
        <0> in tcp_timeouts
) or (
        ADD_TCP_TIMEOUT(21,FTP_CONTROL_TIMEOUT),
        ADD_TCP_TIMEOUT(0,0)
);
```

Code def

```
/*
 * Ansi Spoofing cache update.
* Check if the anti spoofing cache is invalid (Spoof mistmach flags in the
 * packet flags register r_pflags). if it is, update spoof cache field in
 * r_cflags with the current interface (ifid) in the connections table.
 */


ENTRY_SPOOF_CACHE_INVALID(r_pflags),
        ((r_cdir = 1, set r_cflags CHANGE_SPOOF_CACHE_A(r_cflags,ifid),
           ((tcp, modify <conn;r_ckey,r_ctype,r_cflags> in connections) or
                (udp,  modify <conn;0,r_ctype,r_cflags> in connections,
                            modify <udpconn;r_ckey,r_ctype,r_cflags> in connections)))
        or
        (r_cdir = 2, set r_cflags CHANGE_SPOOF_CACHE_B(r_cflags,ifid),
           ((tcp, modify <rconn;r_ckey,r_ctype,r_cflags> in connections) or
                (udp,  modify <rconn;0,r_ctype,r_cflags> in connections,
                      modify <udprconn;r_ckey,r_ctype,r_cflags> in connections)))
         );

/*
 * TCP Connections mechanism
 *
 * There is a good reason why the following rule accepts only established
 * packets.  One might think that the that r_ctype equals  CONN_TCP implies
 * that we have already tested this packet against the rule base,  which is
 * true but not enough.  If the code of the inbound direction  accepts this
 * packet while the code  of the outbound direction rejects it,  the packet
 * is recorded in the connections table  during the inbound rule base pass,
 * and is accepted by this very rule in the outbound pass which is not what
 * the rule base ment.  Checking for established packets will guaranty that
 * the connection will not be established in such cases and the packet will
 * be checked  against the outbound  code as well.
 * Mandatory acceptance of non-established  packets  is  done by  recording
 * the  relvant connection in the  connections table  with IS_ACCEPTED flag
 * set,  in  this case the even SYN packet will be accepted.
 */


accept
        tcp, ENTRY_TYPE(r_ctype) = CONN_TCP,
        (
                established, DUP_EST
        )
```

```
        or
        (
                (r_cdir = 1, ENTRY_ACCEPTED(r_cflags) = IS_ACCEPTED_A)
          or
                (r_cdir = 2, ENTRY_ACCEPTED(r_cflags) = IS_ACCEPTED_B)
          or
                (ENTRY_ACCEPTED(r_cflags) = IS_ACCEPTED_EITHER)
        );


/*
 * UDP Connections mechanism
 */
#ifdef REVERSE_UDP
accept udp, DUP_EST, ENTRY_TYPE(r_ctype) = CONN_UDP, direction = 0 or
        r_cdir = 2 or (r_ctype & _UDP_ESTABLISHED);
#endif


/*
 * SAM code
 * Check the sam table for ipaddrs which are temporarily blocked -
 * such connections will not be allowed to reach the rulebase code.
 */

#define SAM_RULE -3
#define SAM_RESTR_IP            1
#define SAM_RESTR_SRC_IP        2
#define SAM_RESTR_DST_IP        4
#define SAM_RESTR_CONN_SRC 8
#define SAM_RESTR_CONN_DST 16
#define SAM_RESTR_CONN                  32

#define SHORT_NO_ALERT                  1
#define SHORT_ALERT                     2
#define LONG_NO_ALERT           4
#define LONG_ALERT                      8


define SAM_LOG(op, log_mask) {

                (
                        (op = SAM_RESTR_IP, set sr8 log_mask)
                                or
                        (op = SAM_RESTR_SRC_IP, set sr8 log_mask >> 8 )
                                or
```

```
                        (op = SAM_RESTR_DST_IP, set sr8 log_mask >> 16)
                                or
                        (op = SAM_RESTR_CONN, set sr8 log_mask >> 24)
                ),
        (
          ( sr8 & SHORT_NO_ALERT, LOG(short, LOG_NOALERT, SAM_RULE ) )
         or
          ( sr8 & SHORT_ALERT, LOG(short, <"![alert]">, SAM_RULE ) )
         or
          ( sr8 & LONG_NO_ALERT, LOG(long, LOG_NOALERT, SAM_RULE) )
         or
          ( sr8 & LONG_ALERT, LOG(long, <"![alert]">, SAM_RULE) )
         or
           1
        )

};


define SAM_NOTIFY(op, notify_mask) {

          (
                (op = SAM_RESTR_IP, set sr8 notify_mask)
                        or
                (op = SAM_RESTR_SRC_IP, set sr8 notify_mask >> 8)
                        or
                (op = SAM_RESTR_DST_IP, set sr8 notify_mask >> 16)
                        or
                (op = SAM_RESTR_CONN, set sr8 notify_mask >> 24)
          ),
          sr8 & 1 /* If the value isn't 1, that means we're in NOTIFY mode */

};


reject   (
                        set sr7 0,
                        (
                                (set sr9 0, get <sconn> from proxied_conns to sr8,
                                 get sr9 from sam_blocked_ips to sr4) or
                                get origsrc from sam_blocked_ips to sr4 or
                                get xlatesrc from sam_blocked_ips to sr4,
                                set sr7 sr4,
                                ((sr4 & SAM_RESTR_IP, SAM_LOG(SAM_RESTR_IP,
sr5), set sr5 SAM_RESTR_IP)
                                        or
```

```
                                (sr4 & SAM_RESTR_SRC_IP,
SAM_LOG(SAM_RESTR_SRC_IP, sr5), set sr5 SAM_RESTR_SRC_IP))
                        )
                                or
                        (
                                get origdst from sam_blocked_ips to sr4 or
                                get xlatedst from sam_blocked_ips to sr4,
                                set sr7 (sr7 | (sr4<<16)),
                                ((sr4 & SAM_RESTR_IP, SAM_LOG(SAM_RESTR_IP,
sr5), set sr5 SAM_RESTR_IP)
                                        or
                                (sr4 & SAM_RESTR_DST_IP,
SAM_LOG(SAM_RESTR_DST_IP, sr5), set sr5 SAM_RESTR_DST_IP)
                                )
                        )
                                or
                (
                                sr7 & SAM_RESTR_CONN_SRC,  (sr7>>16) &
SAM_RESTR_CONN_DST,
                                (sr9, get <sr9, origdst, origdport, ip_p> from

                sam_blocked_servs to sr5)
                                or
                                        get <origsrc, origdst, origdport, ip_p> from

                sam_blocked_servs to sr5
                                or
                                        get <xlatesrc, xlatedst, xlatedport, ip_p> from

                sam_blocked_servs to sr5
                                SAM_LOG(SAM_RESTR_CONN, sr5), set sr5
SAM_RESTR_CONN
                        )
                ), SAM_NOTIFY(sr5, sr6);


#ifndef NO_ENCRYPTION_FEATURES

#define accept_conn_crypt
                                \
outbound all@all {
                                        \
        ACCEPT_CONN_ENCRYPT;
                                                \
}
                                                        \
```

```
inbound all@all {
                                        \
        ACCEPT_CONN_DECRYPT;
                                                \
}

#define accept_prematch_crypt
                                \
outbound all@all {
                                        \
        ENTRY_PREMATCH_CRYPT(r_pflags),
                                        \
        ACCEPT_CONN_ENCRYPT;
                                                \
}
                                                        \
inbound all@all {
                                        \
        ENTRY_PREMATCH_CRYPT(r_pflags),
                                        \
        ACCEPT_CONN_DECRYPT;
                                                \
}

#else
#define accept_prematch_crypt
#define accept_conn_crypt
#endif

accept_conn_crypt;

/*
 *  This code drops (which makes keepalive) old connections coming in the
 *  reverse direction. This means those connections could only be
 *  re-established in the right direction according to the rule base or
 *  proxied_connections table code. After those are reestablished they would
 *  be in the connections table and accepted there.
 */
eitherbound all@all {
(tcp or udp),<rconn> in old_connections, <conn> not in old_connections,
   ((udp or syn or rst, drop) or
#ifndef NO_ENCRYPTION_FEATURES
          (direction=1, get <rconn> from old_connections to sr1,
           sr1 & CONN_IPSEC, ENTRY_TYPE_OLDCONN(sr1)=CONN_ENC_B,
           (sr2,call KFUNC_TCP_ESTABLISHED<>, ENCRYPT(sr2,sr1),accept ) or
drop)
```

```
        or
#endif
        (call KFUNC_TCP_ESTABLISHED<>, accept)
                or drop);
}
```

crypto def

```
/*
 * RDP Macros
 */
#define RDPPORT                          259
#define RDPCRYPTCMD                      100
#define RDPCRYPT_RESTARTCMD              101
#define RDPUSERCMD                       150
#define RDPSTATUSCMD                 128

/*
 * This macro is used to intercept the first RDP packet in the session
 * in which decryption is requested.
 */
#define RDPCRYPTF
                                         \
                    ((rdp_cmd = RDPCRYPTCMD) or (rdp_cmd = RDPUSERCMD)
\
                    or (rdp_cmd = RDPSTATUSCMD))


/*
 * This macro is used to detect all the other packets in an decryption request
 * session.
 */
#define RDPCRYPT
                                         \
                    (((rdp_cmd & 0x7fffffff) = RDPCRYPTCMD) or
                \
                     ((rdp_cmd & 0x7fffffff) = RDPUSERCMD) or
                \
                          ((rdp_cmd & 0x7fffffff) = RDPSTATUSCMD))


/*
 * This macro is used to intercept the first RDP packet in the session
 * in which encryption-restart is requested.
 */
#define RDPCRYPT_RESTARTF
                                         \
                    (rdp_cmd = RDPCRYPT_RESTARTCMD)
/*
 * This macro is used to detect all the other packets in an encryption-restart
 * request session.
 */
```

```
#define RDPCRYPT_RESTART
                                        \
                ((rdp_cmd & 0x7fffffff ) = RDPCRYPT_RESTARTCMD)


#ifndef NO_ENCRYPTION_FEATURES
/* Non-encryption versions of these follow... */
#define accept_fw1_rdp
                                        \
eitherbound all@all {
                                \
        accept
                                        \
                dport = RDPPORT, udp,
                                        \
                (
                                                \
                        /*
                                                \
                         *    Intercept decryption requests. This rule intercept the      \
                         * first packet in the session. The first packet needs special   \
                         * treatment by the encrypt_intrcpt trap because this GW is not
\
                         * the IP-destination of the packet.
                \
                         *          For SecuRemote, we check whether <srrdpcon> is in the
rdp      \
                         * table.
                                        \
                         *    The packet is held. If the daemon decides to take care of
\
                         * the request, the packet is dropped. Otherwise, it releases   \
                         * the packet.
                                        \
                         */
                                                \
                        direction = 0, RDPCRYPTF,
                                \
                        ((<rdpconn> in rdp_table)
                                \
                                or
                                                \
                        (<srrdpconn> in rdp_table)
                                \
                                or
                                                \
```

```
                (log packet<-1> encrypt_intrcpt, hold))
                \
        ) or (
                                        \
                /*
                                                \
                 * All other RDP packets of the decryption session will be
\
                 * targeted to a specifice host which could be this GW
    \
                 * (on which the inspection is done) or other GWs along the way. \
                 */
                                                \
                RDPCRYPT
                                        \
        ) or (
                                        \
                /*
                                                \
                 * Same as above but for decryption-restart requests. If the RDP \
                 * restart is for a connection that is being encrypted by this   \
                 * side, then we trap the daemon. If not, we just let the packet \
                 * go through to the next firewall.
                \
                 */
                                                \
                direction = 0, RDPCRYPT_RESTARTF,
                        \
                ((get <rdpconn> from connections to r_ckey_ctype_cflags,
\
                        ((ENTRY_TYPE(r_ctype) = CONN_ENC_A)
                                \
                                or
                                                \
                        (ENTRY_TYPE(r_ctype) = CONN_ENC_B)),
                        \
                        log packet<-1> encrypt_intrcpt, drop)
                \
                        or
                                                \
                1)
                                                \
        ) or (
                                        \
                RDPCRYPT_RESTART
                                        \
```

```
                );
                                                                \
}
                                                                \
/*
                                                                \
 *    Reject timed out encryptions (they are moved from the
\
 * encryption_requests table to the rejected_encryptions by the fw daemon,          \
 * when the RDP encryption protocol fails).
                \
 */
                                                                \
eitherbound all@all {
                                        \
        reject
                                                        \
                <conn> in rejected_encryptions;
                                        \
}

/*
 * ENCRYPT/DECRYPT macros
 */
#define ENCRYPT(key,val) (call KFUNC_ENCRYPT <(key),(val),conn>)
#define DECRYPT(key,val) (call KFUNC_DECRYPT <(key),(val)>)

/*
 * How to modify the encryption entry for TCP and UDP connections ?
 */
define tcpudp_calc_entry(type)
        {
                set r_arg 0,
                (
                        (
                                tcp,
                                (
                                        (
                                                /*
                                                 *    In ftp, rsh/rexec and sqlnet sessions,  the
                                                 * sequence and acknowledgment numbers
might change
                                                 * by address translation.
                                                 */
                                                dport = 21 or origdport = 21 or dport = 514
or
```

```
                                        dport = 512 or dport = 1521 or dport = 1525
or
                                        dport = 1526 or dport = 1755 or dport = 554,
                                        set r_entry MATCH_BY_SEQACK_CHG
                                )
                        )
                )
                        or
                (
                        1
                )
        ),
        set r_entry MAKE_ENTRY(type,0,r_entry,r_arg)
};

deffunc RECORD_CONN_ENC(rule,key,connarg) {
        (
                (
                        (rule & 0xffffff00) = 0xffffff00,
                        set sr3 ((sr3 & 0xffff00ff)|((rule & 0x000000ff) << 8)),
                        set sr4 0
                ) or (
                        set sr3 (sr3 | 0x0000ff00),
                        set sr4 rule
                ),
                (<conn> in connections)
                        or
                (<rconn> in connections)
                        or
                (tcp,
                        (
                                tcp_record_ok,
                                (NEED_MORE_INSPECTION(dport),
                                set sr3 (sr3 | MORE_INSPECTION)) or (1),
                                set r_cdir 1,
                                set r_ckey DUP_KEY(key),
                                set r_ctype connarg,
                                set r_cflags (sr3|SPOOF_CACHE_EMPTY),
                                record
<conn;r_ckey,r_ctype,r_cflags@TCP_START_TIMEOUT>

                                in connections
                        )
                )
                                                or
                (udp,
```

```
                        (
                                packetlen >=28,
                                (NEED_MORE_INSPECTION(dport),
                                set sr3 (sr3 | MORE_INSPECTION)) or (1),
                                set r_cdir 1,
                                set r_ckey key,
                                set r_ctype (_UDP_ESTABLISHED | connarg),
                                set r_cflags (sr3|SPOOF_CACHE_EMPTY),
                                UDP_RECORD(conn,0,r_ctype,r_cflags),
                                (
                                        <udpconn> in connections
                                                        or
                                        (
                                                direction = 1 or set r_ctype connarg,

        UDP_RECORD(udpconn,DUP_KEY(r_ckey),r_ctype,r_cflags)
                                        )
                                )
                        )
                )
        )
};
```

Request a TCP or a UDP Encryption

We assume that the code of this macro works under a drop action and hence, if we return 1 and do not change the action, the packet will be dropped.

In order to prevent excessive encryption requests for the same connection due to retransmissions, we record each encryption reque st in a table (called 'encryption_requests' due to some strange coincidence), and at the beginning of the process we check if the current connection is in that table, and if so, we ignore the packet, return 1 and therefore drop it.

We also consider here the case of 'close' or 'related' UDP `connections', which means that if we have an encrypted UDP connection from port A of host X to port B of host Y, we accept and encrypt any UDP connection from port A of host X to host Y. This is for cases like RPC.

When we trap the daemon, we hold the packet in the kernel and pass the id of the packet to the encryption invocation mechanism (to see how it is done, look at the format of encrypt_invoke which is defined in traps.def). Then, the daemon starts the encryption protocol which may, as you probably expect, either succeed or fail. If it succeeds, the daemon deletes the correspondent from the encryption_requests table and stores the information about this new encrypted connection in the connections table. If the

protocol fails, the daemon moves the correspondent record from the encryption_requests table to the rejected_encryptions one. In both cases, the daemon releases the packet which the kernel holds, which causes the packet to be rescanned through the inspect code, so the packet is either encrypted or rejected.

```
define DUP_UDP_CONN     {
        (
                tcpudp_calc_entry(CONN_ENC_A),
                set r_entry CHANGE_TYPE(r_entry,ENTRY_TYPE(r_ctype)),
                set r_ckey DUP_KEY(r_ckey),
                UDP_RECORD(conn,0,r_entry,0),
                UDP_RECORD(udpconn,r_ckey,r_entry,0),
                (
                        (
                                ENTRY_TYPE(r_entry)= CONN_ENC_A, direction = 1,
                                ENCRYPT(r_ckey,r_entry)
                        ) or (
                                ENTRY_TYPE(r_entry)= CONN_ENC_B, direction = 0,
                                DECRYPT(r_ckey,r_entry)
                        ) or 1
                )
        )
};

define REQUEST_ENCRYPTION_TCPUDP(rule)  {
        (
                (
                        (
                                <conn> in encryption_requests
                                                        or
                                <rconn> in encryption_requests
                        ), vanish
                ) or (
                        udp,
                                get <udpconn> from connections to r_ckey_ctype_cflags,
                                DUP_UDP_CONN,
                                accept
                ) or (
                        tcpudp_calc_entry(CONN_ENC_A),
                        record <conn> in encryption_requests,
                        (NEED_MORE_INSPECTION(dport) or sr3 > 0,
                    record <conn> in crypt_more_inspection) or (1),
                        set sr1 rule,
                        log encrypt_invoke,
                        hold
                )
```

```
        )
};

/*
 * Encrypt what needed. In SecuRemote - Update enc_timer table.
 */
define ACCEPT_CONN_ENCRYPT() {
        accept
                (
                        ENTRY_TYPE(r_ctype) = CONN_ENC_A,
                        (
                                (r_cdir=2) // accept outgoing packets which were already
decrypted
                                 or
#ifdef SECUREMOTE
                                ((ENCRYPT(r_ckey,r_ctype), record 1 in enc_timer) or
drop)
#else
                                (ENCRYPT(r_ckey,r_ctype) or drop)
#endif
                        )
                )
                or
                (
                        ENTRY_TYPE(r_ctype) = CONN_ENC_B,
                        (
                                (r_cdir=1) // accept outgoing packets which were already
decrypted
                                 or
#ifdef SECUREMOTE
                                ((ENCRYPT(r_ckey,r_ctype), record 1 in enc_timer) or
drop)
#else
                                (ENCRYPT(r_ckey,r_ctype) or drop)
#endif
                        )
                );
};

/*
 * Decrypt what needed. In SecuRemote - Update enc_timer table.
 */
define ACCEPT_CONN_DECRYPT() {
        accept
                (
                        ENTRY_TYPE(r_ctype) = CONN_ENC_A,
```

```
                        (
                                (r_cdir = 1) // accept incoming packets which should be
encrypted
                                or
#ifdef SECUREMOTE
                                ((DECRYPT(r_ckey,r_ctype), record 1 in enc_timer) or
drop)
#else
                                (DECRYPT(r_ckey,r_ctype) or drop)
#endif
                        )
                )
                or
                (
                        ENTRY_TYPE(r_ctype) = CONN_ENC_B,
                        (
                                (r_cdir = 2) // accept incoming packets which should be
encrypted
                                or
#ifdef SECUREMOTE
                                ((DECRYPT(r_ckey,r_ctype), record 1 in enc_timer) or
drop)
#else
                                (DECRYPT(r_ckey,r_ctype) or drop)
#endif
                        )
                );
};
```

Encrypt a TCP or a UDP Encryption

We assume that the code of this macro works under a drop action and hence, if we return 1 and do not change the action, the packet will be dropped.

Actually, most encryptions (and decryptions) are done by the code which is in the code.def file. This macro is attached to the code which is generated from a specific rule and it does two things.

First, a daemon who is asked to decrypt some connection, doesn't store the information about that connection in the connections table (upon successful negotiation), but rather in the decryption_pending table. By this we enforce that the first packet of an encrypted connection will be matched against the rulebase. For that, upon receiving the first packet of the connection, this macro calls the Daemon to move the current connection from the pending table to the connections table, while verifying that the encryption schemes which were selected by both daemons matches those, specified by the rule.

Then, if the current connection is not in pending table, this macro starts an encryption protocol, using the REQUEST_ENCRYPTION_TCPUDP macro, described above.
*/

```
define ENCRYPTION_TCPUDP(rule)        {
        (
#ifdef TCP_FASTMODE_ACTIVE
                udp or (NOT_TCP_FASTMODE_PORT(dport,rule),
                                NOT_TCP_FASTMODE_PORT(sport,rule)),
#endif
                (
                        <origconn,rule> in trapped
                )
                                        or
                (
                        get <origconn> from decryption_pending to sr1,
                        set r_ckey SAFE_DUP_KEY(sr1),
                        delete <origconn> from decryption_pending,
                        tcpudp_calc_entry(CONN_ENC_B),
                        record <origconn,rule> in trapped,
                        (NEED_MORE_INSPECTION(dport) or sr3 > 0,
                         record <conn> in crypt_more_inspection) or (1),
                        set sr1 rule, log decrypt_invoke,
                        hold
                )
                                        or
                (
                        REQUEST_ENCRYPTION_TCPUDP(rule)
                )
        )
};
```

Request a Non-TCP/UDP Encryption

We assume that the code of this macro works under a drop action and hence, if we return 1 and do not change the action, the packet will be dropped.

In order to prevent excessive encryption requests for the same connection due to retransmissions, we record each encryption request in a table (called 'encryption_requests' due to some strange coincidence), and at the beginning of the process we check if the current connection is in that table, and if so, we ignore the packet, return 1 and therefore drop it.

Unlike TCP and UDP encryption, here we do not have connections (or pseudo connections) so what we enecrypt all the communication between certain two hosts which a key, associated with the ip protocol and the rule number. This implies that instead of using the classic tuple <src,sport,dst,dport,ip_p>, we have to use <src,rule-

number,dst,0,ip_p>. This also means that if we are* encryption a full duplex protocol, we will have two encryption requests, two keys and two entries: one for packets going from A to B and one for packets going from B to A.

Before trapping the daemon, we check if <src,rule-number,dst,0,ip_p> is in the connections table. If so, we consider four cases:

Side	Direction	Action
A	Incoming	Accept
A	Outgoing	Encrypt
B	Incoming	Decrypt
B	Outgoing	Accept

When we trap the daemon, we hold the packet in the kernel and pass the id of the packet to the encryption invocation mechanism (to see how it is done, look at the format of encrypt_invoke which is defined in traps.def). Then, the daemon starts the encryption protocol which may, as you probably expect, either succeed or fail. If it succeeds, the daemon deletes the correspondent from the encryption_requests table and stores the information about this new encrypted <src,rule-number,dst,0,ip_p> channel in the connections table. If the encryption protocol fails then the daemon moves the correspondent record from the encryption_requests table to the rejected_encryptions one. In both cases, the daemon releases the packet which the kernel holds, which causes the packet to be rescanned through the inspect code, so the packet is either encrypted or rejected.

```
define ENCRYPTION_OTHER(rule)          {
        (
                (
                        /* Accept this packet if the 'accepted' flags is set in the
                           connections table. */
                        get <src,rule,dst,0,ip_p> from connections to r_ckey_ctype_cflags,
                        (ENTRY_ACCEPTED(r_cflags), accept) or
                        ((
                                (
                                        direction = 0, ENTRY_TYPE(r_ctype) =
CONN_ENC_B,
                                        DECRYPT(r_ckey,r_ctype)
                                )
                                or
                                (
                                        direction = 1, ENTRY_TYPE(r_ctype) =
CONN_ENC_A,
                                        ENCRYPT(r_ckey,r_ctype)
                                )
                                or
                                        (1)
```

```
                        ), accept)
                ) or (
                        <origsrc,rule,origdst,0,ip_p,rule> in trapped
                ) or (
                        get <origsrc,0,origdst,0,ip_p> from decryption_pending to sr1,
                        set r_ckey SAFE_DUP_KEY(sr1),
                        /*
                         *  Note that we do not delete the request from
decryption_pending.
                         * this is because we want to handle retransmissions correctly.
                         */
                        tcpudp_calc_entry(CONN_ENC_B),
                        record <origsrc,rule,origdst,0,ip_p,rule> in trapped,
                        ((icmp, set r_connarg ((icmp_type << 8) | icmp_code)) or 1),
                        set sr1 rule, log decrypt_invoke_other,
                        hold
                ) or (
                          /*
                           * Ignore this packet if a similar packet is being treated
                           * already.
                           */
                        <src,rule,dst,0,ip_p> in encryption_requests
                ) or (
                        record <src,rule,dst,0,ip_p> in encryption_requests,
                        ((icmp, set r_connarg ((icmp_type << 8) | icmp_code)) or 1),
                        set sr1 rule,
                        log encrypt_invoke,
                        hold
                )
        )
};
```

Decryption accept macro

```
deffunc ENCRYPTION(rule) {
        (
                ( (wasskipped, record <src, dst, rule> in first_packet_was_decrypted) or
1),
                ( (wasskipped, call KFUNC_IPPOOL_ALLOCATE <conn> or drop)
                        or 1),
                (tcp or udp),
                get <src,dst,rule> from FW_IPSEC_SHORTCUT_TABLE_ID to sr11,
                (<src, dst, rule> in first_packet_was_decrypted, set sr13 CONN_ENC_B)
                        or
                (set sr13 CONN_ENC_A),
```

```
                RECORD_CONN_ENC(rule, sr11, sr13),
                ( (sr12, call KFUNC_LOG_ENCRYPTION_CONN
<wasskipped,sr11,rule>) or 1),
                (direction = 0, ACCEPT_CONN_DECRYPT)
                        or
                (ACCEPT_CONN_ENCRYPT)
        ) or (
                (tcp or udp), (ENCRYPTION_TCPUDP(rule) or 1)
        ) or (
                ENCRYPTION_OTHER(rule) or 1
        )
};
```

User client encryption (SecuRemote) macro (server side):

The user should first successfully pass the key exchange protocol, and as a result his IP address will be placed in userc_rules (<src,rule>).

The first packet from the user will be intercepted by this rule and if the source and this rule (<src,rule>) or the destination (<dst,0>) are in userc_rules, then this connection is trapped (the daemon still has to check that the destination (or source, if <dst,0> was found in userc_rules) is in the encryption domain of the gateway). If the connection should be decrypted (encrypted), the daemon adds the connection to the connections table (with its key, of course).
In any case the packet is released.

```
define USER_DECRYPTION_OTHER(rule) {
        (
          (<src,rule,dst,0,ip_p> in decryption_pending, drop) or
                (
                        get <src,rule,dst,0,ip_p> from connections to r_ckey_ctype_cflags,
                        ENTRY_TYPE(r_ctype) = CONN_ENC_B,
                        DECRYPT(r_ckey,r_ctype),accept
                ) or (
                        record <src,rule,dst,0,ip_p> in decryption_pending,
                        ((icmp, set r_connarg ((icmp_type << 8) | icmp_code)) or 1),
                        set sr1 rule, set sr2 1,
                        log userc_server_invoke,
                        hold, drop
                )
    )
};

define USER_ENCRYPTION_OTHER(rule) {
        (
                (<src,rule,dst,0,ip_p> in decryption_pending, drop) or
```

```
                (
                        get <src,rule,dst,0,ip_p> from connections to r_ckey_ctype_cflags,
                        ENTRY_TYPE(r_ctype) = CONN_ENC_A,
                        ENCRYPT(r_ckey,r_ctype),accept
                ) or (
                        record <src,rule,dst,0,ip_p> in decryption_pending,
                        ((icmp, set r_connarg ((icmp_type << 8) | icmp_code)) or 1),
                        set sr1 rule, set sr2 0,
                        log userc_server_invoke,
                        hold, drop
                )
        )
};

USER_DENCRYPTION_TCPUDP handles both encryption and decryption for tcp and
udp connections */

define USER_DENCRYPTION_TCPUDP(rule, is_decrypt)        {
        (
                (<conn> in decryption_pending, drop) or
                (record <conn> in decryption_pending,
                 set sr1 rule, set sr2 is_decrypt,
                 (NEED_MORE_INSPECTION(dport) or sr3 > 0,
                  record <conn> in crypt_more_inspection) or (1),
                 log userc_server_invoke,
                 hold, drop)
        )
};

define USER_ENCRYPTION(rule) {
        (direction = 1,
                (not(get <src> from userc_dont_trap to sr1) or sr1),
                (
                        (
                                (tcp or udp), USER_DENCRYPTION_TCPUDP(rule,0)
                        ) or (
                                USER_ENCRYPTION_OTHER(rule)
                        )
                )
        )
};

if the ip protocol is not tcp or udp, USER_DECRYPTION accepts the packet if it is
outbound and was previously decrypted by this rule (on the other interface) */

define USER_DECRYPTION(rule,intersect) {
```

```
        (
                (direction = 0,
                        ( not(get <dst> from userc_dont_trap to sr2) or
                (sr2, (not(rule) or not(intersect)))),
                        (
                                (
                                        (tcp or udp),
USER_DENCRYPTION_TCPUDP(rule,1)
                                ) or (
                                        USER_DECRYPTION_OTHER(rule)
                                )
                        )
                )
                or
                (direction = 1, not(tcp or udp),
                 <src,rule,dst,0,ip_p> in connections, accept)
        )
};

deffunc USER_CLIENT_ENCRYPTION(rule) {
        (
                (<src,rule> in userc_rules,
                 set sr1 userc_rules[src,rule],USER_DECRYPTION(rule,sr1))
        )
};
```

USERC_DECRYPT_SRC checks whether the connection should be decrypted. The services appearing should not be decrypted even if the source is a client that has exchanged keys with the gateway and the destination is in the encryption domain.
The following macro should be the 'not' of the accept_without_encryption macro defined in CLCRYPT definition.

```
define USERC_DECRYPT_SRC {
        (
#ifndef ENCDNS
                not(dport = SERV_domain, (udp or tcp)),
#endif
#ifdef SECUREMOTE
                not(<ip_p,dport> in userc_noncrypt_ports)
#else
                not(dport = FWD_TOPO_PORT, tcp),
                not(dport = FWD_SVC_PORT, tcp),
                not(dport = FWM_SVC_PORT, tcp),
                not(dport = ISAKMPD_DPORT, udp or tcp),
                not(_fwz_encapsulation),
```

```
                not(_esp),
                not(_ah)
#endif
        )
};

deffunc ACCEPT_CLIENT_ENCRYPTION(rule) {
        (
                USERC_DECRYPT_SRC,
                (direction = 0, <src,0> in userc_rules,
                 USER_DECRYPTION(rule,0))
                or
                (direction = 1, <dst,0> in userc_rules, USER_ENCRYPTION(rule))
        )
};
```

USERC_CHECK checks whether the connection may match the client encryption rule. This macro does not perform any action. It is assumed that this macro appears in rules where trapping to the daemon and encryption/decryption will be handled by other macros (such as rules with resources)

```
define USERC_CHECK(rule) {
        (<src,rule> in userc_rules)
};
```

ACCEPT_DECRYPT allows an incoming encrypted packet to be decrypted and then accepted if it matches an accept rule and is found in decryption_pending. In the "other" case (i.e. not tcp or udp) we also need to check the connections table and call the macro ENCRYPTION_OTHER if required.

```
deffunc ACCEPT_DECRYPT(rule) {
        (
                (
                        tcp or udp, <origconn> in decryption_pending,
                        (ENCRYPTION_TCPUDP(rule) or 1)
                )
                                or
                (
                        (
                                <origsrc, 0, origdst, 0, ip_p> in decryption_pending
                                                                        or
                                <origsrc, rule, origdst, 0, ip_p> in connections
                        ), (ENCRYPTION_OTHER(rule) or 1)
                )
                                or
                (
```

```
                        ((WAS_DECRYPTED) or wasskipped) ,
                        ENCRYPTION(rule) or 1
                )
        )
};

deffunc IS_NOT_DECRYPTED(rule) {
        not (
                (
                        tcp or udp, <origconn> in decryption_pending
                ) or (
                        not(tcp or udp),
                        (
                                <origsrc, 0, origdst, 0, ip_p> in decryption_pending
                        ) or (
                                <origsrc, rule, origdst, 0, ip_p> in connections
                        )
                ) or (
                        (WAS_DECRYPTED) or wasskipped
                )
        )
};

#else /* NO_ENCRYPTION_FEATURES */

/* non-encryption: Do this to let RDP through on eitherbound */
#define accept_fw1_rdp
                                        \
eitherbound all@all {
                                \
        accept
                                        \
                dport = RDPPORT, udp,
                                        \
                RDPCRYPTF or RDPCRYPT or RDPCRYPT_RESTARTF or
RDPCRYPT_RESTART;               \
}

#define ENCRYPTION(rule) (1)
#define USER_CLIENT_ENCRYPTION(rule) (1)

#endif /* NO_ENCRYPTION_FEATURES */

#endif /* __crypt_def__ */
```

Basic definitions

All the below are definitions used as standard in every rulebase. Whenever you define fx. FTP there is a macro defined below that is called in case such packet comes.

Some common definitions

```
// #define FTP_NON_STANDARD

#ifdef FWEMBEDED
#define NO_XLATION
#endif

#defineany 1
#define other 1

#defineconn src,sport,dst,dport,ip_p
#define rconn dst,dport,src,sport,ip_p
#definesconn 0,sport,dst,dport,ip_p
#define udpconn src,sport,dst,0,ip_p
#define udprconn dst,dport,src,0,ip_p
#define rdpconn  cr_src,cr_sport,cr_dst,cr_dport,cr_p
#define srrdpconn sr_cr_src, sr_cr_dst
#define rdprconn cr_dst,dr_dport,cr_src,cr_sport,cr_p
#define srrdpconn sr_cr_src, sr_cr_dst
#define origconn origsrc,origsport,origdst,origdport,ip_p
#define icmptcpudprconn
icmp_ip_src,icmp_th_sport,icmp_ip_dst,icmp_th_dport,icmp_ip_p
#define icmptcpudpconn
icmp_ip_dst,icmp_th_dport,icmp_ip_src,icmp_th_sport,icmp_ip_p



#define inbound =>
#define outbound <=
#define eitherbound <>
```

Basic TCP/IP Macros

Due to a yet unrecovered bug in FW-1 INSPECT compiler, do not try to use something like !tcp. It will not work.

```
define SERV_ftp { ftp };
define SERV_shell { shell };
define SERV_exec { exec };
define SERV_sunrpc { sunrpc };
define SERV_domain { 53 };
```

```
define SERV_snmp { 161 };
define SERV_rip { 520 };
define PROTO_tcp { tcp };
define PROTO_udp { udp };
define PROTO_icmp { icmp };
define PROG_portmap { 100000 };

define tcp { ip_p = tcp };
define udp { ip_p = udp };
define icmp { ip_p = icmp };
define rpc { any };
#define _icmp icmp_Is_Obsolete.__Use_icmp-proto.
```

routing protocols

```
define _ggp { ip_p = 3 };
define _egp { ip_p = 8 };
define _igrp { ip_p = 9 };
define _hello { ip_p = 63 };
define _ospf { ip_p = 89 };

// offset of first tcp data byte
#define TCPDATA (20 + (([32:1] >> 2) & 0xfc))
#define UDPDATA    28
```

icmp redirect

```
define icmp_redirect { icmp, icmp_type = ICMP_REDIRECT };
define icmp_no_redirect { icmp, icmp_type != ICMP_REDIRECT };
#define icmp_request  (icmp_type in icmp_requests)
#define icmp_reply            (icmp_type in icmp_replies)
#define icmp_error            (icmp_type in icmp_errors)
```

firewall ports

```
#ifndef FWD_SVC_PORT
#define FWD_SVC_PORT 256
#define FWD_LOG_PORT 257
#defineFWD_SNMP_PORT 260
#define FWD_TOPO_PORT 264
#define FWD_KEY_PORT 265
#endif

#ifndef FWM_SVC_PORT
#define FWM_SVC_PORT 258
#endif
```

ISAKMP daemon ports

```
#define ISAKMPD_SPORT 500
```

```
#define ISAKMPD_DPORT 500
```

Encryption IP protocols

```
define _ah              { ip_p = 51 };
define _esp             { ip_p = 50 };
define _fwz_encapsulation  { ip_p = 94 };

// ports which are dangerous to connect to
define NOTSERVER_TCP_PORT(p) {
                (not
                        (
                                ( p in tcp_services, set sr10 RCODE_TCP_SERV, set sr11
0,
                                  set sr12 p, set sr1 0, log bad_conn)
                        or
                                ( p < 1024, set sr10 RCODE_SMALL_PORT, set sr11 0,
set sr12 p,
                                  set sr1 0, log bad_conn)
                        )
                )
};

define NOTSERVER_UDP_PORT(p) {
                (not
                        (
                                ( p in udp_services, set sr10 RCODE_UDP_SERV, set sr11
0,
                                  set sr12 p, set sr1 0, log bad_conn)
                        or

                         ( p < 1024, set sr10 RCODE_SMALL_PORT, set sr11 0,
                                  set sr12 p, set sr1 0, log bad_conn)
                        )
                )
};

define NOT_TCP_FASTMODE_PORT(p,r) {
#ifdef TCP_FASTMODE_ACTIVE
                (not (p in tcp_fastmode_services, set sr10
RCODE_TCP_FASTMODE_PORT,
                         set sr11 0, set sr12 p, set sr1 r, log bad_conn))
#else
                (1)
#endif
};
```

```
#define WRONG_HOST_LOG
                                                        \
  (set sr1 0, set sr10 RCODE_WRONG_HOST, set sr11 0, set sr12 0, log bad_conn)

#define NULL_MACRO(rule)  (1)                        /* used by code generation */

#ifndef NO_ENCRYPTION_FEATURES
#define DUP_KEY(key) (call KFUNC_KBUF_DUP <key>)
#define SAFE_DUP_KEY(key) (call KFUNC_KBUF_SAFE_DUP <key>)
#else
#define DUP_KEY(key) (0)
#define SAFE_DUP_KEY(key) (0)
#endif

#define IS_MY_IPADDR(addr) (call KFUNC_IS_MY_IPADDR <addr>)

#defineNEED_MORE_INSPECTION(dport)
                                                \
          (dport in prolog_services or origdport in prolog_services)

#ifndef TCP_CTRLS
#define TCP_CTRLS  (CTRL_SRC_FIN | CTRL_DST_FIN |
CTRL_TCP_ESTABLISHED)
#endif

#define TABLE_NOT_EMPTY(table) ((table > 31) or ((r_tab_status >> table) & 1))

#define accept_fwz_as_clear(type)
                                \
          (
                                                                \
                    ((ENTRY_TYPE(type) = CONN_TCP) or (ENTRY_TYPE(type) =
CONN_UDP) or        \
                    (call KFUNC_NOT_IPSEC_ENCAPS  <r_ckey>), accept) or 1
                    \
          )
```

fetch the flag registers from the connections table. This is done for every packet before the INSPECT gets the packet, however for connections that 'prolog' INSPECT code puts in the connectios tables we update these flags here.

```
#define update_tcp_flags(type,dir)                                  \
```

```
    (set r_cdir (dir),
                        \
                        ((dir) = 1, get <conn> from connections to r_ckey_ctype_cflags)
        \
                        or
                                                \
                        (get <rconn> from connections to r_ckey_ctype_cflags))

#define update_udp_flags(type,dir)                          \
                (set r_cdir (dir),
                                        \
                        ((dir)=1, get <udpconn> from connections to
r_ckey_ctype_cflags)\
                        or
                                                \
                        (get <udprconn> from connections to r_ckey_ctype_cflags))

#define accept_tcp_noncrypt(type,dir)
                        \
        (
                                                \
                update_tcp_flags(type,dir),
                                \
                (ENTRY_TYPE(type) = CONN_TCP, accept)
                                \
                or
                                                \
                (set r_pflags (r_pflags | PREMATCH_CRYPT))
                        \
        )

#define accept_udp_noncrypt(type,dir)
                        \
        (
                                                \
                update_udp_flags(type,dir),
                                \
                (ENTRY_TYPE(type) = CONN_UDP, accept)
                                \
                or
                                                \
                (set r_pflags (r_pflags | PREMATCH_CRYPT))
                        \
        )
```

for services using KFUNC_XLATE_ANTICIPATE.

- num2 is the direction of the direction of the packet in respect to the control connection.
- num1 is for changing only a certain offset in the packet:
  1. means according to source,
  2. to destination.

```
#define BUILD_CONT_REV(num1,num2)
                                \
        ((num1 << 4) + num2)
```

TCP / UDP Virtual Connections Mechanism

```
#define UDP_RECORD(con,key,type,flags)
                        \
        (record <con;key,type,flags|SPOOF_CACHE_EMPTY@UDP_TIMEOUT> in
connections)

#define tcp_record_ok
                                        \
        (packetlen >= 40, (syn or ((th_flags & (TH_URG | TH_RST)) = 0)))

#ifndef NO_ENCRYPTION_FEATURES
#define WAS_ENCRYPTED
                                        \
        (
                                                \
                (
                                                \
                        (set sr1 ENTRY_TYPE_OLDCONN(old_connections[conn]), sr1)
        \
                                                                or
                                                \
                        (set sr1 ENTRY_TYPE_OLDCONN(old_connections[rconn]),
sr1)            \
                ), (sr1 = CONN_ENC_A or sr1 = CONN_ENC_B)
                        \
        )

#define WAS_DECRYPTED
                                        \
        (
                                                \
                (ENTRY_TYPE_OLDCONN(old_connections[conn])) = CONN_ENC_B
                \
        )
```

```
#else
#define WAS_ENCRYPTED 0
#define WAS_DECRYPTED 0
#endif

#include "crypt.def"

deffunc RECORD_CONN(rule) {
#ifdef TCP_FASTMODE_ACTIVE
        (tcp, dport in tcp_fastmode_services)
                or
#endif
        (
                (
                        (rule & 0xffffff00) = 0xffffff00,
                        set sr3((sr3 & 0xffff00ff)|((rule & 0x000000ff) << 8)),
                        set sr4 0
                ) or (
                        set sr3 (sr3 | 0x0000ff00),
                        set sr4 rule
                ),
#ifndef NO_ENCRYPTION_FEATURES
                ACCEPT_CLIENT_ENCRYPTION(sr4)
                        or
#endif
                (<conn> in connections)
                        or
                (<rconn> in connections)
                        or
#ifndef NO_ENCRYPTION_FEATURES
#if ACCEPT_DECRYPT_ENABLE
                (direction = 0, ACCEPT_DECRYPT(sr4))
                                                or
                (
#else
                (IS_NOT_DECRYPTED(sr4),
#endif
#else
                (
#endif
                        (tcp,
                                (
                                        established, WAS_ENCRYPTED, drop
                                ) or (
                                        tcp_record_ok,
```

```
                                        NOT_TCP_FASTMODE_PORT(sport,sr4) or
reject,
                                        (NEED_MORE_INSPECTION(dport),
                                        set sr3 (sr3 | MORE_INSPECTION)) or (1),
                                        set r_cdir 1,
                                        set r_ckey 0,
                                        set r_ctype CONN_TCP,
                                        set r_cflags (sr3|SPOOF_CACHE_EMPTY),
                                        record
<conn;r_ckey,r_ctype,r_cflags@TCP_START_TIMEOUT>

                                in connections
#ifdef SECUREMOTE
                                        ,USERC_RECORD_VERIFIED_CONN
#endif
                                )
                        )
                                                        or
                        (udp,
                                (
                                        WAS_ENCRYPTED, drop
                                ) or (
                                        packetlen >=28,
                                        (NEED_MORE_INSPECTION(dport),
                                        set sr3 (sr3 | MORE_INSPECTION)) or (1),
                                        set r_cdir 1,
                                        set r_ckey 0,
                                        set r_ctype (CONN_UDP |
_UDP_ESTABLISHED),
                                        set r_cflags (sr3|SPOOF_CACHE_EMPTY),
                                        UDP_RECORD(conn,r_ckey,r_ctype,r_cflags),
                                        (
                                                <udpconn> in connections
                                                        or
                                                (
                                                        direction = 1 or set r_ctype
CONN_UDP,

        UDP_RECORD(udpconn,r_ckey,r_ctype,r_cflags)
                                                )
                                        )
#ifdef SECUREMOTE
                                        ,USERC_RECORD_VERIFIED_CONN
#endif
                                )
                        )
```

```
                                                    or
                                                (1)
                )
        )
};
```

Accounting Macros

```
#ifndef NO_ACCOUNT
/*
 * This function should be called when a connection is matched to the 'magic'
 * entry (in special protocols such as ftp-PASV, VDOLive, etc.).
 * It replaces the entry with one that has the actual port number where the
 * 'magic' was before.
 */
deffunc ACCOUNT_MATCH(smagic,save,timeout) {
        (
                (
                        get <0,src,smagic,dst,dport,ip_p> from tracked to sr1,
                        record <0,conn; sr1,sr2,sr3,sr4,sr5 @timeout> in tracked,
                        (save or delete <0,src,smagic,dst,dport,ip_p> from tracked)
                 )
                 or 1
         )
};

deffunc RACCOUNT_MATCH(smagic,save,timeout) {
        (
                (
                        get <0,dst,dport,src,smagic,ip_p> from tracked to sr1,
                        record <0,rconn; sr1,sr2,sr3,sr4,sr5 @timeout> in tracked,
                        (save or delete <0,dst,dport,src,smagic,ip_p> from tracked)
                 )
                 or 1
         )
};


#else
#define ACCOUNT_MATCH(smagic,save,timeout) 1
#define RACCOUNT_MATCH(smagic,save,timeout) 1
#endif

/* Services code outside base.def */
```

```
#include "xtreme.def"
#include "dcerpc.def"
#include "h323.def"
```

Remote Procedure Call (Sun RPC)

```
#define RPC_RESTRICTIVE_REQUESTS
#define RPC_RESTRICTIVE_GETPORT
#define RPC_RESTRICTIVE_CALLIT
/*#define RPC_RESTRICTIVE_DELCONN*/
/*#define ALLOW_RPC_GETTIME*/

#ifdef pm_prog
#undef pm_prog
#undef pm_prot
#endif

#define rm_ansport [52, b]

#define rpc_cred_len            [(UDPDATA+28),b]
#define rpc_ver_len             [(UDPDATA+36+rpc_cred_len),b]
#define pm_prog
        [(UDPDATA+40+rpc_cred_len+rpc_ver_len),b]
#define pm_prot
        [(UDPDATA+48+rpc_cred_len+rpc_ver_len),b]
#define pm_protoname            [(UDPDATA+52+rpc_cred_len+rpc_ver_len),b]

#define rm_xid_tcp              [TCPDATA+4,b]
#define rm_direction_tcp        [TCPDATA+8,b]
#define rm_ansport_tcp          [TCPDATA+28,b]
#define cb_prog_tcp             [TCPDATA+16,b]
#define cb_vers_tcp             [TCPDATA+20,b]
#define cb_proc_tcp             [TCPDATA+24,b]
#define pm_prog_tcp             [TCPDATA+44,b]
#define pm_protoname_tcp        [TCPDATA+56,b]

#define PMAPPROC_GETPORT 3
#define PMAPPROC_CALLIT  5

#define RPCPORT_VAL             (sr1.[0:1])
#define RPCPORT_NEXT            (set sr1 (sr1 + 1))

#define RPC_GETBYTE
                                \
```

```
        (
                                                        \
                set sr2 RPCPORT_VAL - 0x30,
                                        \
                RPCPORT_NEXT, ((RPCPORT_VAL != 0x2e, RPCPORT_VAL !=
0x00,                   \
                set sr2 ((sr2 * 10) + (RPCPORT_VAL - 0x30)),
                \
                RPCPORT_NEXT, (RPCPORT_VAL != 0x2e, RPCPORT_VAL != 0x00,
                \
                set sr2 ((sr2 * 10) + (RPCPORT_VAL - 0x30)), RPCPORT_NEXT) or 1)
or 1),\
                RPCPORT_NEXT
                                                \
        )

#define RPC_GETADDR(hostreg,portreg,offset)
                        \
        (
                                                        \
                set sr1 (offset),
                                        \
                RPC_GETBYTE, set hostreg sr2 << 24,
                                \
                RPC_GETBYTE, set hostreg (hostreg | (sr2 << 16)),
                \
                RPC_GETBYTE, set hostreg (hostreg | (sr2 << 8)),
        \
                RPC_GETBYTE, set hostreg (hostreg | sr2),
                \
                RPC_GETBYTE, set portreg sr2 << 8,
                                \
                RPC_GETBYTE, set portreg (portreg | sr2)
                \
        )

define rpc_pmap_reply() {
        accept
                r_cdir = 2, sport = SERV_sunrpc,
                (udp, rm_direction = 1,         set sr1 rm_xid)
#ifdef RPC_PORTMAPPER_OVER_TCP
                  or
                (tcp, rm_direction_tcp = 1, set sr1 rm_xid_tcp)
#endif  /* RPC_PORTMAPPER_OVER_TCP */
                ,(
                        get <dst,src,ip_p,dport,sr1> from pmap_req to sr5,
```

```
                        (
                                (sr7 = PROTO_udp)
#ifdef RPC_OVER_TCP
                                  or
                                (sr7 = PROTO_tcp)
#endif  /* RPC_OVER_TCP */
                        ),
                        (
                                sr6 = 2,
                                (udp, set sr4 rm_ansport)
#ifdef RPC_PORTMAPPER_OVER_TCP
                                  or
                                (tcp, set sr4 rm_ansport_tcp)
#endif  /* RPC_PORTMAPPER_OVER_TCP */
                        ) or (
                                sr6 = 3,
                                (udp, RPC_GETADDR(sr3,sr4,(UDPDATA+28)))
#ifdef RPC_PORTMAPPER_OVER_TCP
                                  or
                                (tcp, RPC_GETADDR(sr3,sr4,(TCPDATA+32)))
#endif  /* RPC_PORTMAPPER_OVER_TCP */
                                ,sr3 = src or reject
                        ),
                        sr4, record <src,sr7,sr4;sr5> in rpc_serv,
                        delete <dst,src,ip_p,dport,sr1> from pmap_req
                )
#ifndef NO_ENCRYPTION_FEATURES
                ,(ACCEPT_CLIENT_ENCRYPTION(0) or 1);
#endif
};

define rpc_pmap_request_udp() {
#ifdef RPC_RESTRICTIVE_REQUESTS
        r_cdir = 1, dport = SERV_sunrpc, udp, <udpconn> in rpc_sessions,
        (
                cb_proc = PMAPPROC_GETPORT
#ifdef RPC_RESTRICTIVE_GETPORT
                ,pm_prog = rpc_sessions[udpconn]
#endif
        ) or (
                cb_proc = PMAPPROC_CALLIT
#ifdef RPC_RESTRICTIVE_CALLIT
                ,pm_prog = rpc_sessions[udpconn]
#endif
        ) or (
#ifdef RPC_RESTRICTIVE_DELCONN
```

```
                delete <conn> from connections, delete <udpconn> from connections,
                delete <udpconn> from rpc_sessions,
#endif
                LOG(long, LOG_NOALERT, 0) ,drop
        );
#else
        1
#endif
};

define rpc_pmap_request_tcp() {
#ifdef RPC_PORTMAPPER_OVER_TCP
        (
                r_cdir = 1, dport = SERV_sunrpc, tcp, rm_direction_tcp = 0,
                cb_prog_tcp = PROG_portmap,
                (
                        cb_proc_tcp = PMAPPROC_GETPORT,
                        (cb_vers_tcp = 2, set sr1 pm_prot)
                          or
                        (cb_vers_tcp = 3,
                                (pm_protoname_tcp = 0x75647000, set sr1 17)
        /* UDP */
                                  or
                                (pm_protoname_tcp = 0x74637000, set sr1 6)
        /* TCP */
                        ),
                        (
                                (sr1 = PROTO_udp)
#ifdef RPC_OVER_TCP
                                  or
                                (sr1 = PROTO_tcp)
#endif  /* RPC_OVER_TCP */
                        ),
                        record <src,dst,ip_p,sport,rm_xid_tcp;pm_prog_tcp,

        cb_vers_tcp,sr1> in pmap_req
#ifdef ALLOW_RPC_GETTIME
                ) or (cb_proc_tcp = 6
#endif /* ALLOW_RPC_GETTIME */
                ) or (
#ifdef RPC_RESTRICTIVE_DELCONN
                        delete <conn> from connections,
#endif
                        LOG(long, LOG_NOALERT, 0) ,reject
                )
        )
```

```
#else /* RPC_PORTMAPPER_OVER_TCP */
        1
#endif /* RPC_PORTMAPPER_OVER_TCP */
};

#define rpc_code
                                        \
        rpc_pmap_reply;
                                                \
        rpc_pmap_request_udp;
        rpc_pmap_request_tcp;

#ifndef PMAP_CONNECT_TIMEOUT
#define PMAP_CONNECT_TIMEOUT 30
#endif

deffunc rpc_insession(rpcnum) {
        (
                (
                        ((cb_prog = rpcnum), udp)
#ifdef RPC_OVER_TCP
                  or
                        (first, tcp)
#endif  /* RPC_OVER_TCP */
                ),
                (
                        (
                                not <dst,ip_p,dport> in rpc_serv,
                                not <dst> in rpc_serv_hosts,
                                not <dst> in pmap_not_responding,
                                TRAP_TO(pmap_fetch, PMAP_FETCH,
PMAP_CONNECT_TIMEOUT), hold, 0
                        )
                        or
                        (
                                rpc_serv[dst,ip_p,dport] = rpcnum,
                                set r_entry MATCH_BY_RPC,  set r_connarg rpcnum
                        )
                )
        )
};

/*
 * Change from define to deffunc because pm_prog passed as param from auth.def
 */
deffunc rpc_getport(rpcnum) {
```

```
	(
		dport = SERV_sunrpc,
		(
			udp, rm_direction = 0, cb_prog = PROG_portmap,
			(
				cb_proc = PMAPPROC_GETPORT, pm_prog = rpcnum,
				(cb_vers = 2, set sr10 pm_prot)
				  or
				(cb_vers = 3,
					(pm_protoname = 0x75647000, set sr10 17)  /*
UDP */
					  or
					(pm_protoname = 0x74637000, set sr10 6)
	/* TCP */
				),
				(
					(sr10 = PROTO_udp)
#ifdef RPC_OVER_TCP
					  or
					(sr10 = PROTO_tcp)
#endif  /* RPC_OVER_TCP */
				),
				record <src,dst,ip_p,sport,rm_xid;rpcnum,cb_vers,sr10>

				in pmap_req,
				record <udpconn; rpcnum> in rpc_sessions,
				set r_entry MATCH_BY_GETPORT, set r_connarg
rpcnum
			)
		)
#ifdef RPC_PORTMAPPER_OVER_TCP
		or tcp
#endif  /* RPC_PORTMAPPER_OVER_TCP */
	)
};


deffunc rpc_callit(rpcnum) {
	(
		dport = SERV_sunrpc, udp, rm_direction = 0, cb_prog = PROG_portmap,
		(
			cb_proc = PMAPPROC_CALLIT, pm_prog = rpcnum,
			record <udpconn; rpcnum> in rpc_sessions,
			set r_entry MATCH_BY_CALLIT, set r_connarg rpcnum
		)
	)
```

```
};

deffunc rpc_prog(rpcnum) {
        rpc_getport(rpcnum)
  or
        rpc_insession(rpcnum)
  or
        rpc_callit(rpcnum)
};
```

NIS+ support

```
#define align4(x)       ((x+3) & 0xfffffffc)
#define PROG_nisplus 100300
#define NISPLUSPROC_IBLIST 5

#define NISPLUS_MAGIC 0x4e495350

#define nisplus_prolog1
                                        \
        (
                                                \
                cb_prog_tcp = PROG_nisplus, tcp,
                        \
                rpc_serv[dst,ip_p,dport]=cb_prog_tcp,
                \
                cb_proc_tcp = NISPLUSPROC_IBLIST,
                        \
                set sr1 32 + align4(([TCPDATA+32,b])) + 8,          /* Credentials */
        \
                set sr1 sr1 + align4(([TCPDATA+sr1,b])) + 4,/* Verifier     */
        \
                set sr1 sr1 + align4(([TCPDATA+sr1,b])) + 4,/* Name         */
                \
                (
                                                \
                        [TCPDATA+sr1,b],                                        /*
Attributes */           \
                        set sr1 sr1 + 4,                                /* Attr No. */
                \
                        set sr1 sr1 + align4(([TCPDATA+sr1,b])) + 4, /* Attr Name */
        \
                        set sr1 sr1 + align4(([TCPDATA+sr1,b])) + 4  /* Attr Value */
        \
```

```
                ) or (
                                                        \
                        set sr1 sr1 + 4                                         /* Attr
No. */                   \
                ),
                                                        \
                set sr1 sr1 + 8,                                        /* Flags +
Object */
#define nisplus_prolog2
                                        \
                [TCPDATA+sr1,b],                                                /*
Callbacks */                    \
                set sr1 sr1 + 4,                                        /* Callback
No. */           \
                set sr1 sr1 + align4(([TCPDATA+sr1,b])) + 4,/* Callback Hostname */
        \
                set sr1 sr1 + 8,                                        /*
No.Addresses + Len */\
                RPC_GETADDR(sr3,sr4,(TCPDATA+sr1)),
                                        \
                sr3 = src or reject, NOTSERVER_TCP_PORT(sr4) or reject,
                \
                record <src, sr4, dst, NISPLUS_MAGIC, PROTO_tcp;
DUP_KEY(r_ckey),   \

        r_ctype,r_cflags> in pending, \
    (ENTRY_TRACKED(r_cflags),
                                \
                record <0,src,sr4,dst,NISPLUS_MAGIC,PROTO_udp;conn
@PENDING_TIMEOUT>\

                                in tracked)     \
                or 1,
                                                \
                accept_fwz_as_clear(r_ctype)
                        \
        )

#define nisplus_prolog nisplus_prolog1 nisplus_prolog2;

#define nisplus_prematch
                                \
        (
                                                        \
                tcp, first, TABLE_NOT_EMPTY(pending),
                        \
```

```
                get <dst,dport,src,NISPLUS_MAGIC,ip_p> from pending to sr10,
        \
                set sr5 CHANGE_MATCH(sr11,MATCH_BY_PROTOCOL),
                                \
                set sr5 REMOVE_CTRL(sr5, TCP_CTRLS),
                                \
                record <rconn;DUP_KEY(sr10),sr5,IS_ACCEPTED_B
|TRACKED_TRANS(sr12)|   \

        SPOOF_CACHE_EMPTY@7200> in connections, \
                direction = 0
                                        \
                or delete <dst,dport,src,NISPLUS_MAGIC,ip_p> from pending,
        \
                (ENTRY_TRACKED(sr12),
                                        \
                RACCOUNT_MATCH(NISPLUS_MAGIC,0,TCP_TIMEOUT)) or 1,
                                \
                accept_tcp_noncrypt(sr5,2)
                                \
        );
```

File Transfer Protocol (FTP) - Implicit Connections

```
/* Match flag for KFUNC_FTPPORT */
#define FTPPORT_MATCH (0x2)

#define IS_PORT_CMD (([TCPDATA,b] & 0xdfdfdfdf) = 0x504f5254)

#define IS_PASV_MSG ([TCPDATA,b] = 0x32323720) /* 227 (pasv message number)
*/

#define FTP_MAGIC 0x46545053        /* "FTPS" */

#ifdef FTP_NON_STANDARD
#define FTP_MAGIC2          0x46545050        /* "FTPP" */
#endif

#define FTPPORT(match)     (call KFUNC_FTPPORT <0x1|(match)>)

//
//    Use this if you do not want the FW-1 module to insist on a newline at the
// end of the PORT command:
```

```
// #define FTPPORT(match)  (call KFUNC_FTPPORT <(match)>)

#define FTP_ENFORCE_NL


//    ftpdata_code records  outgoing ftp PORT requests  and accepts its related
// incoming responds. When the FTP connection is encrypted, its keys are copied
// to the data connection to come.

#ifdef NO_XLATION
#define FTPPORT_ANTICIPATE(port)  1
#define FTPPASV_ANTICIPATE(port)  1
#else
#define FTPPORT_ANTICIPATE(port)
                                \
                (call KFUNC_XLATE_ANTICIPATE
                                        \
                                <0, (port), dst, 0, PROTO_tcp, BUILD_CONT_REV(0,0),
                \
                                        PENDING_TIMEOUT, 1, 0, 0>)

#define FTPPASV_ANTICIPATE(port)
                                \
                (call KFUNC_XLATE_ANTICIPATE
                                        \
                                <dst, 0, 0, (port), PROTO_tcp, BUILD_CONT_REV(0,1),
                \
                                        PENDING_TIMEOUT, 1, 0, 0>)

#endif /* NO_XLATION */

/*
 * FTP_TRACK_DATA_CONN
 * For accounting of data connections, we record a translation entry in the
 * 'tracked' table.
 * This entry is of the form: <0, data-conn; control-conn> . Before doing this,
 * we check to see if the ifid value of the control connection is -1.  If it
 * is, we know the connection was entered through the ftp auth daemon, in
 * which case we do not want the translation entry, since the data connection
 * will be counted separately on its own entry, entered by the auth daemon.
 * we put TRACK_UNKNOWN in r_cflags because in authenticated ftp the data
 * connections can be tracked as normal connection or as translated connection
 */
#define FTP_TRACK_DATA_CONN(s_port, d_port, timeout)
                \
```

```
                ((ENTRY_TRACKED(r_cflags),
                                        \
                        (r_cdir = 1, get <conn> from tracked to sr2, sr7 != -1,
        \
                         record <0,src,s_port,dst,d_port,ip_p;conn @timeout> in tracked) \
                        or
                                                        \
                        (get <rconn> from tracked to sr2, sr7 != -1,
        \
                         record <0,dst,d_port,src,s_port,ip_p;rconn@timeout> in tracked) \
                )
                                                        \
                or 1)

/*
 * flags in the ftp_restrictions table:
 * 1 - enable PORT   2 - enable PASV   4 - enable bidirectional data conns
 */

#ifdef ALL_MODULES_4_1_SP2_OR_ABOVE
#define NL_AT_END
                                        \
        (set sr4 call KFUNC_FIND_STR<TCPDATA,1,0x0a000000>, sr4 = (packetlen -
1))
#else
#define NL_AT_END
                                        \
        ([packetlen - 1:1] = 0x0a)
#endif

#ifdef FTP_ENFORCE_NL
#define FTP_CHECK_COMMAND
                                        \
        (
                                                        \
                (packetlen - TCPDATA) = 0
                                \
                                        or
                                                        \
                ENTRY_TYPE(r_ctype) = CONN_ENC_A
                                \
                                        or
                                                        \
                ENTRY_TYPE(r_ctype) = CONN_ENC_B
                                \
```

```
                                        or
                                                \
                NL_AT_END
                                        \
        )
#else
#define FTP_CHECK_COMMAND (1)
#endif

#define ftp_accept_port_clear(port,bidi)
                \
                ENTRY_TYPE(r_ctype) = CONN_TCP,
                                        \
                <src,port,dst,20,ip_p> in connections or
                \
                record <src,port,dst,20,ip_p;0,CONN_TCP, IS_ACCEPTED_B |
        \
                 SPOOF_CACHE_EMPTY | bidi |
                                        \
                        (TRACKED_TRANS(r_cflags) & TRACK_UNKNOWN)
|RECORD_SRC(0x82)>       \

                        in connections,         \
                FTP_TRACK_DATA_CONN(port, 20, TCP_TIMEOUT)

#define ftp_accept_port_enc(port,bidi)
                        \
                (
                                                \
                        ENTRY_TYPE(r_ctype) = CONN_ENC_A
                                \
                                        or
                                                \
                        ENTRY_TYPE(r_ctype) = CONN_ENC_B
                                \
                ),
                                                \
                set sr2 CHANGE_MATCH(r_ctype,MATCH_BY_PROTOCOL),
                        \
                set sr2 REMOVE_CTRL(sr2, TCP_CTRLS),
                \
                record <src,port,dst,20,ip_p;DUP_KEY(r_ckey),sr2,
                \
                  bidi|SPOOF_CACHE_EMPTY|
                                        \
```

```
                    (TRACKED_TRANS(r_cflags) &
TRACK_UNKNOWN)|RECORD_SRC(0x83)>                \

                        in connections,         \
                FTP_TRACK_DATA_CONN(port, 20, TCP_TIMEOUT)

#ifndef FTP_NON_STANDARD

#define ftp_accept_port
                                        \
                r_cdir = 1, dport = SERV_ftp or origdport = SERV_ftp, tcp,
        \
                FTP_CHECK_COMMAND or reject,
                                        \
                IS_PORT_CMD, set sr1 FTPPORT(0),
                                \
                (get <conn> from ftp_restrictions to sr2, sr2 & 1 or reject,            \
                 (sr2 & 4, set sr2 NO_CONN_ONEWAY) or set sr2
CONN_ONEWAY_EITHER) or\
                set sr2 CONN_ONEWAY_EITHER,
                                        \
                direction = 1 or FTPPORT_ANTICIPATE(sr1),
                        \
                set sr1 FTPPORT(FTPPORT_MATCH), sr1 != 0 or
(WRONG_HOST_LOG,reject),\
                NOTSERVER_TCP_PORT(sr1) or reject,
                                \
                direction = 0 or FTPPORT_ANTICIPATE(sr1),
                        \
                (
                                                        \
                        ftp_accept_port_enc(sr1,sr2)
                        \
                ) or (
                                                \
                        ftp_accept_port_clear(sr1,sr2)
                                \
                ),
                                                        \
                accept_fwz_as_clear(r_ctype)

#else
/* replacing the old ftp_accept_port version with a new one which insert to pending table
packets with FTP Client port command properties.
*/
```

```
#define ftp_accept_port
                                        \
                r_cdir = 1, tcp,
                \
                dport = SERV_ftp or origdport = SERV_ftp,
                \
                FTP_CHECK_COMMAND or reject, IS_PORT_CMD,
                                        \
                set sr1 FTPPORT(0),                                                     \
                (get <conn> from ftp_restrictions to sr2, sr2 & 1 or reject,            \
                (sr2 & 4, set sr2 NO_CONN_ONEWAY) or set sr2
CONN_ONEWAY_EITHER) or \
                set sr2 CONN_ONEWAY_EITHER,
                        \
                direction = 1 or FTPPASV_ANTICIPATE(sr1),
                        \
                set sr1 FTPPORT(FTPPORT_MATCH), sr1 != 0 or
(WRONG_HOST_LOG,reject), \
                NOTSERVER_TCP_PORT(sr1) or reject,
                                        \
                direction = 0 or FTPPASV_ANTICIPATE(sr1),
                        \
                record <dst,FTP_MAGIC2,src,sr1,ip_p;
                        \
                                                DUP_KEY(r_ckey),r_ctype,r_cflags|sr2> in
pending,                                \
                FTP_TRACK_DATA_CONN(sr1, FTP_MAGIC2,
PENDING_TIMEOUT),                                       \
                accept_fwz_as_clear(r_ctype)
#endif


#define ftp_record_pasv
                                                \
                r_cdir = 2, tcp,
                                        \
                sport = SERV_ftp or sport = auth_services[SERV_ftp,ip_p],
        \
                FTP_CHECK_COMMAND or reject,
                                                \
                IS_PASV_MSG, set sr1 FTPPORT(0),
                                        \
                (get <rconn> from ftp_restrictions to sr2, sr2 & 2 or reject,           \
                 (sr2 & 4, set sr2 NO_CONN_ONEWAY) or set sr2
CONN_ONEWAY_EITHER) or\
```

```
                 set sr2 CONN_ONEWAY_EITHER,
                                        \
                direction = 1 or FTPPASV_ANTICIPATE(sr1),
                        \
                set sr1 FTPPORT(FTPPORT_MATCH), sr1 != 0 or
(WRONG_HOST_LOG,reject),\
                NOTSERVER_TCP_PORT(sr1) or reject,
                                \
                direction = 0 or FTPPASV_ANTICIPATE(sr1),
                        \
                record
<dst,FTP_MAGIC,src,sr1,ip_p;DUP_KEY(r_ckey),r_ctype,r_cflags|sr2>\

                                in pending,     \
                FTP_TRACK_DATA_CONN(sr1, FTP_MAGIC,
PENDING_TIMEOUT),                               \
                accept_fwz_as_clear(r_ctype)


#define ftp_accept_pasv1
                                \
                tcp, first, TABLE_NOT_EMPTY(pending),
                        \
                get <src,FTP_MAGIC,dst,dport,ip_p> from pending to sr10,
        \
                set sr5 CHANGE_MATCH(sr11,MATCH_BY_PROTOCOL),
                                \
                set sr5 REMOVE_CTRL(sr5, TCP_CTRLS),
#define ftp_accept_pasv2
                                \
                NOT_TCP_FASTMODE_PORT(sport,0) or reject,
                        \
                set sr2 (sr12 & CONN_ONEWAY_EITHER),
                                \
                <conn> in connections or
                                \
                record <conn;DUP_KEY(sr10),sr5,IS_ACCEPTED_A|sr2|
                \
                                SPOOF_CACHE_EMPTY|(TRACKED_TRANS(sr12) &
TRACK_UNKNOWN)|         \

        RECORD_SRC(0x84)> in connections,       \
      delete <src,FTP_MAGIC,dst,dport,ip_p> from pending,
        \
                (ENTRY_TRACKED(sr12),
                                        \
```

```
                ACCOUNT_MATCH(FTP_MAGIC,0,TCP_TIMEOUT)) or 1,
                                \
                accept_tcp_noncrypt(sr11,1)

#define ftp_accept_pasv ftp_accept_pasv1 ftp_accept_pasv2


/* ftp_accept_serv
New service for Server's random data connection ports. This service is similiar to FTP
passive data connections. A new FTP_MAGIC number - FTP_MAGIC2 has been defined
to avoid conflict with preceding packets in pending table with the same FTP_MAGIC
number. first - SYN TCP packet without an ACK
*/

#ifdef FTP_NON_STANDARD
#define ftp_accept_serv1
                                \
            tcp, first, TABLE_NOT_EMPTY(pending),
        \
                get <src,FTP_MAGIC2,dst,dport,ip_p> from pending to sr10,
        \
                set sr5 CHANGE_MATCH(sr11,MATCH_BY_PROTOCOL),
                                \
                set sr5 REMOVE_CTRL(sr5, TCP_CTRLS),

#define ftp_accept_serv2
                                \
                NOT_TCP_FASTMODE_PORT(sport,0) or reject,
                        \
                set sr2 (sr12 & CONN_ONEWAY_EITHER),
                                \
                <conn> in connections or
                                \
              record <rconn;DUP_KEY(sr10),sr5,
                \

        IS_ACCEPTED_B|sr2|CONN_ONEWAY_EITHER|SPOOF_CACHE_EMPTY|
        \
                                        (TRACKED_TRANS(sr12) &
TRACK_UNKNOWN)|RECORD_SRC(0x84)> \
                                in connections,
                                                \
      delete <src,FTP_MAGIC2,dst,dport,ip_p> from pending,
        \
            (ENTRY_TRACKED(sr12),
                        \
```

```
                RACCOUNT_MATCH(FTP_MAGIC2,0,TCP_TIMEOUT)) or 1,
                                \
                accept_tcp_noncrypt(sr11,2)

#define ftp_accept_serv ftp_accept_serv1 ftp_accept_serv2
#endif

#define ftpdata_code ftp_accept_port
```

Remote Shell (RSH) - Implicit Connections and Remote Exec (REXEC) - Implicit Connections

```
#define RSH_MAGIC          0x52534820          /* "RSH " */
#define REXEC_MAGIC 0x45584543                 /* "EXEC" */

#define RSHPORT_5
                                        \
        ((([TCPDATA:1]   - 48) * 10000) +
                        \
         (([TCPDATA+1:1] - 48) *  1000) + (([TCPDATA+2:1] - 48) * 100) +
        \
         (([TCPDATA+3:1] - 48) *    10) + (([TCPDATA+4:1] - 48) *   1))
#define RSHPORT_4
                                        \
        ((([TCPDATA:1]   - 48) *  1000) + (([TCPDATA+1:1] - 48) * 100) +
        \
         (([TCPDATA+2:1] - 48) *    10) + (([TCPDATA+3:1] - 48) *   1))
#define RSHPORT_3
                                        \
        ((([TCPDATA:1]   - 48) *   100) + (([TCPDATA+1:1] - 48) *  10) +
        \
         (([TCPDATA+2:1] - 48) *     1))

#define RSH_PORT
                                        \
                (
                                                \
                        ((packetlen - TCPDATA) = 6, set sr1 RSHPORT_5)
                        \
                                                        or
                                                \
                        ((packetlen - TCPDATA) = 5, set sr1 RSHPORT_4)
                        \
```

```
                                                        or
                                                  \
                    ((packetlen - TCPDATA) = 4, set sr1 RSHPORT_3)
                    \
            )

#ifndef NO_XLATION
#define RSH_ANTICIPATE(port)
                                \
        (call KFUNC_XLATE_ANTICIPATE
                                        \
                    <0, (port), dst, 0, PROTO_tcp, BUILD_CONT_REV(0,0),
            \
                                PENDING_TIMEOUT, 2, 0, 0>)
#else
#define RSH_ANTICIPATE(port) 1
#endif


#define RSH_RECORD_PENDING(port,magic)
                                \
        (record <src,port,dst,(magic),ip_p;DUP_KEY(r_ckey),r_ctype,r_cflags>  \

                                in pending,     \
   (ENTRY_TRACKED(r_cflags),
                                \
         record <0,src,port,dst,(magic),ip_p;conn @PENDING_TIMEOUT> in tracked)\
        or 1)


// record outgoing rsh sessions in pending table

#define rsh_table_store
                                \
            (dport = SERV_shell, set sr1 RSH_MAGIC)
                    \
                                        or
                                                  \
            (dport = SERV_exec,  set sr1 REXEC_MAGIC), tcp, first,
        \
            record <0,src,sport,dst,sr1,(th_seq+1);0,0,0> in pending

// extract stderr port number from a first packet of a recorded rsh session
// and store in pending
```

```
#define rsh_table_update
                                        \
                r_cdir = 1, TABLE_NOT_EMPTY(pending),
                                        \
                (dport = SERV_shell, set sr2 RSH_MAGIC)
                        \
                                        or
                                                        \
                (dport = SERV_exec,  set sr2 REXEC_MAGIC),
                        \
                tcp, established, (packetlen - TCPDATA) > 0, RSH_PORT,
        \
                get <0,src,sport,dst,sr2,th_seq> from pending to sr3,
        \
                (sr5 = 0 or sr5 = sr1),
                                        \
                record <0,src,sport,dst,sr2,th_seq;0,0,sr1> in pending,
        \
                (sr2 = RSH_MAGIC, sr1 > 600, sr1 < 1024)
                        \
                                                        or
                                                        \
                (sr2 = REXEC_MAGIC, NOTSERVER_TCP_PORT(sr1) or reject),
                        \
                RSH_RECORD_PENDING(sr1,sr2),
                                        \
                RSH_ANTICIPATE(sr1),
                                        \
                accept_fwz_as_clear(r_ctype)

// allow an incoming rsh stderr session if recorded in pending, record the
// connection in the connections table with IS_ACCEPTED_B and CONN_ONEWAY_B
// flags set.

#define rsh_table_test1
                                        \
                tcp, first,
                                        \
                TABLE_NOT_EMPTY(pending),
                                        \
                (sport < 1024, set sr1 RSH_MAGIC)
                        \
                                        or
                                                        \
                (sport not in tcp_services, set sr1 REXEC_MAGIC),
                \
```

```
                get <dst,dport,src,sr1,ip_p> from pending to sr10

#define rsh_table_test2
                                        \
                set sr6 CHANGE_MATCH(sr11, MATCH_BY_PROTOCOL),
                                \
                set sr6 REMOVE_CTRL(sr6, TCP_CTRLS),
        \
                NOT_TCP_FASTMODE_PORT(sport,0) or reject,
                        \
                record <rconn;DUP_KEY(sr10),sr6, IS_ACCEPTED_B |
CONN_ONEWAY_B |         \

        TRACKED_TRANS(sr12)|SPOOF_CACHE_EMPTY|RECORD_SRC(0x86)>
        \

                        in connections,         \
                delete <dst,dport,src,sr1,ip_p> from pending,
                \
                (ENTRY_TRACKED(sr12),
                                        \
                RACCOUNT_MATCH(sr1,0,TCP_TIMEOUT)) or 1,
                                \
                accept_tcp_noncrypt(sr11,2)

#define rsh_table_test rsh_table_test1, rsh_table_test2;

#define rshstderr_code rsh_table_store; rsh_table_update; rsh_table_test;
```

VDOLive

```
#define VDOLive                 7000

#define VDO_HEADER(offset)
                                        \
                        ([TCPDATA+(offset):4,b] = 0x56444f20,
                        \
                         [TCPDATA+(offset+4):4,b] = 0x4c697665)

#define VDO_PORT(offset)
                                \
                        (TCPDATA+(offset)+8)

#define VDO_MAGIC       0x56444f4c              /* "VDOL" */
```

```
#ifndef NO_XLATION
#define VDO_ANTICIPATE(port,offset)
                                        \
          (call KFUNC_XLATE_ANTICIPATE
                                                \
           <0, (port), dst, 0, PROTO_udp, BUILD_CONT_REV(0,0),
PENDING_TIMEOUT,          \
           0, VDO_PORT(offset)+2, 0>)
#else
#define VDO_ANTICIPATE(port,offset) 1
#endif

#define vdolive_intercept1
                                        \
                    r_cdir = 1, dport = VDOLive, tcp, packetlen >= 76, VDO_HEADER(24),
          \
                    set sr1 [VDO_PORT(24):4,b],NOTSERVER_UDP_PORT(sr1) or reject,
          \
                    VDO_ANTICIPATE(sr1,24),
                                                \
                    record
<src,sr1,dst,VDO_MAGIC,PROTO_udp;DUP_KEY(r_ckey),r_ctype,       \

          r_cflags> in pending,  \
      (ENTRY_TRACKED(r_cflags),
                                        \
                    record <0,src,sr1,dst,VDO_MAGIC,PROTO_udp;conn
@PENDING_TIMEOUT>    \

                                        in tracked) \
      or 1,
                                                \
                    accept_fwz_as_clear(r_ctype)

#define vdolive_intercept2
                                        \
                    r_cdir = 1, dport = VDOLive, tcp, packetlen >= 60, VDO_HEADER(8),
          \
                    set sr1 [VDO_PORT(8):4,b],NOTSERVER_UDP_PORT(sr1) or reject,
          \
                    VDO_ANTICIPATE(sr1,8),
                                                \
                    accept_fwz_as_clear(r_ctype)

#define vdolive_accept1
                                                \
```

```
                udp,
                                                \
                TABLE_NOT_EMPTY(pending),
                                        \
                get <dst,dport,src,VDO_MAGIC,ip_p> from pending to sr10,
        \
                (
                                                \
                        (ENTRY_TYPE(sr11) = CONN_TCP, set sr5 CONN_UDP)
                                \
                                                or
                                                \
                        (set sr5 ENTRY_TYPE(sr11))
                                        \
                ),
#define vdolive_accept2
                                        \
                (ENTRY_TRACKED(sr12),
                                        \
                RACCOUNT_MATCH(VDO_MAGIC,0,UDP_TIMEOUT)) or 1,
                                \
                UDP_RECORD(rconn,0,sr5 | _UDP_ESTABLISHED,
                                \

        TRACKED_TRANS(sr12)|RECORD_SRC(0xa1)),          \
                UDP_RECORD(udprconn,DUP_KEY(sr10),sr5 |
_UDP_ESTABLISHED,                       \
                                CONN_ONEWAY_B | TRACKED_TRANS(sr12) |
RECORD_SRC(0xa1)),      \
                delete <dst,dport,src,VDO_MAGIC,ip_p> from pending,
                \
                accept_udp_noncrypt(sr5,2)

#define vdolive_prolog vdolive_intercept1; vdolive_intercept2
#define vdolive_prematch vdolive_accept1 vdolive_accept2
```

Real Audio

```
/*
 * the format of RAUDIO packets is:
 * "PNA" string(0x504e41) (3 bytes) + protocol version (2 bytes) + series of
 * startup messages.
 * Startup messages have the format:
 * Message Identifier (2 bytes) + Message Length (2 bytes) + n bytes of data,
 *   where n == the message length.
```

```
 * Message ID 1 is UDP Requested option, which as long as the version is
 * less than 256 has the following format:
 *  | 0 |  1 |  0 |  2 | p1 | p2 |
 *  | UDP Req | UDPR Len| UDP Port|
 *
 * e.g. a packet could have the format "PNA050102nn00"
 */

#define RAudio                  7070

#define RA_HEADER               ([TCPDATA,b] = 0x504e4100)

#define RA_MSB                  sr1.[0:1]
#define RA_LSB                  sr1.[1:1]
#define RA_LENMSB               sr1.[2:1]
#define RA_LENLSB sr1.[3:1]
#define RA_PORT                 (((sr1.[4:1]) << 8) + sr1.[5:1])
#define RA_NEXT                 (set sr1 (sr1 + ((RA_LENMSB) << 8) + RA_LENLSB +
4))
#define RA_ONOTEND              (RA_LSB != 0 or RA_MSB != 0)
#define RA_OPORT  (RA_LSB = 1, RA_MSB = 0)
#define RA_CHOPT  (RA_ONOTEND, RA_NEXT, RA_OPORT)
#define RA_MAGIC  0x52415544                    /* "RAUD" */
#define RA_FLOWMAGIC 0x52415546                         /* "RAUF" */
#define RA_PENDING_TIMEOUT 900

#define RA_SCAN
                                                \
                (
                                                \
                        set sr1 TCPDATA + 5,
                                        \
                        (
                                                \
                                RA_OPORT or
                                                \
                                RA_CHOPT or
                                                \
                                RA_CHOPT or
                                                \
                                RA_CHOPT
                                                \
                        )
                                                \
                )
```

```
#define RA_FLOWSCAN             ([TCPDATA,b] = 0x4f080007)
#define RA_FLOWPORT             ([(TCPDATA+4):2,b])

#ifndef NO_XLATION
#define RA_ANTICIPATE(port,offset)
                                \
                (call KFUNC_XLATE_ANTICIPATE
                                \
                 <0, (port), dst, 0, PROTO_udp, BUILD_CONT_REV(0,0),
        \
                                PENDING_TIMEOUT, 0, (offset), 0>)
#else
#define RA_ANTICIPATE(port,offset) 1
#endif

#define raudio_intercept
                                \
                (r_cdir = 1, dport = RAudio, tcp,
                        \
                RA_HEADER, RA_SCAN, set sr2 RA_PORT, set sr3 RA_MAGIC,
                        \
                NOTSERVER_UDP_PORT(sr2) or reject,
                                \
                RA_ANTICIPATE(sr2,sr1+4)) or
                                \
                (r_cdir = 2, sport = RAudio, tcp,
                        \
                RA_FLOWSCAN, set sr2 RA_FLOWPORT, set sr3 RA_FLOWMAGIC,
                        \
                NOTSERVER_UDP_PORT(sr2) or reject),
                                \
                record <src,sr2,dst,sr3,PROTO_udp;DUP_KEY(r_ckey),r_ctype,
        \

        r_cflags@RA_PENDING_TIMEOUT> in pending, \
      (ENTRY_TRACKED(r_cflags),
                                \
                record <0,src,sr2,dst,sr3,PROTO_udp;conn
@RA_PENDING_TIMEOUT>                    \

                                in tracked)     \
                or 1,
                                                \
                accept_fwz_as_clear(r_ctype)
```

```
#define raudio_accept1
                                        \
                udp,
                                                \
                TABLE_NOT_EMPTY(pending),
                                        \
                (get <dst,dport,src,RA_MAGIC,ip_p> from pending to sr10,
        \
                (
                                                \
                        (ENTRY_TYPE(sr11) = CONN_TCP, set sr5 CONN_UDP)
                                \
                                                or
                                                \
                        (set sr5 ENTRY_TYPE(sr11))
                                        \
                ),
                                                \
                (ENTRY_TRACKED(sr12),
                                        \
                RACCOUNT_MATCH(RA_MAGIC,0,UDP_TIMEOUT)) or 1,
                                \
                UDP_RECORD(rconn,0,sr5 | _UDP_ESTABLISHED,
                                \

        TRACKED_TRANS(sr12)|RECORD_SRC(0xa3)), \
                UDP_RECORD(udprconn,DUP_KEY(sr10),sr5 |
_UDP_ESTABLISHED,                       \

        TRACKED_TRANS(sr12)|RECORD_SRC(0xa3)), \
                delete <dst,dport,src,RA_MAGIC,ip_p> from pe nding,
                \
                accept_udp_noncrypt(sr5,2)) or
#define raudio_accept2
                                        \
                (get <dst,dport,src,RA_FLOWMAGIC,ip_p> from pending to sr10,
        \
                (
                                                \
                        (ENTRY_TYPE(sr11) = CONN_TCP, set sr5 CONN_UDP)
                                \
                                                or
                                                \
                        (set sr5 ENTRY_TYPE(sr11))
                                        \
```

```
                ),
                                                        \
                (ENTRY_TRACKED(sr12),
                                        \
                ACCOUNT_MATCH(RA_FLOWMAGIC,0,UDP_TIMEOUT)) or 1,
                                \
                /* Record only full connection to avoid overriting the partial
        \
                  connection */
                                        \
                UDP_RECORD(conn,0,sr5 | _UDP_ESTABLISHED,
                                \

        TRACKED_TRANS(sr12)|RECORD_SRC(0xa5)), \
                delete <dst,dport,src,RA_MAGIC,ip_p> from pending,
                \
                accept_udp_noncrypt(sr5,1))

#define raudio_prolog  raudio_intercept
#define raudio_prematch raudio_accept1 raudio_accept2
```

RTSP

```
/*
 * Protocol Description:
 * client starts TCP connection on port 554.
 * client sends "port" command to server, then server starts UDP on that port
 * to client, there might be a few UDP data connections on one session.
 * Inspect Description:
 * RTSP_intercept_client looks for a SETUP Command on a client packet, that
 * contains the port/s of UDP connection to follow. It saves the expected
 * UDP port in rtsp_tab - to allow it.
 * RTSP_intercept_server looks for a packet with RTSP header and then looks
 * for the server_port and a confirmation on the client_port.
 * the connections are removed from the connection table after 900 seconds.
 */
#ifndef NO_XLATION
#define RTSP_ANTICIPATE(p_offset, port)
                        \
                (call KFUNC_XLATE_ANTICIPATE
                                        \
                 <0, (port), dst, 0, PROTO_udp, BUILD_CONT_REV(0,0),
        \
                                PENDING_TIMEOUT, 2, (p_offset), 0>)
#else
```

```
#defineRTSP_ANTICIPATE(port,p_offset) 1
#endif


#define RTSP                        554
#defineRTSP_TIMEOUT                 900
#define RTSP_SETUP_HEADER           ([TCPDATA,b] = 0x53455455) /*"SETU"*/
#define RTSP_HEADER                         ([TCPDATA,b] = 0x52545350) /*"RTSP"*/
#define RTSP_CL_STR                         0x636c6965,0x6e745f70,0x6f72743d /*
"client_port=" */
#define RTSP_SRV_STR                0x73657276,0x65725f70,0x6f72743d /*
"server_port=" */
#define RTSP_CL_STR_LEN                     12
#define RTSP_SRV_STR_LEN            12


#define RTSP_UDP_RECORD(rtsrc,rtsport,rtdst,rtdport,rtipp,key,type,flags)       \
        (record <rtsrc,rtsport,rtdst,rtdport,rtipp;key,type,
        \
                        flags|SPOOF_CACHE_EMPTY@RTSP_TIMEOUT> in
connections)

#define RTSP_XLATE_CHECK_PORTS(addr, port)
                                \
        (not
                                        \
                (
                                                \
                        <addr, PROTO_udp, port> in FWX_ALLOC_TABLE_ID,
                        \
                        <addr, PROTO_udp, port + 1> in FWX_ALLOC_TABLE_ID
                        \
                )
                                                \
        )

deffunc RTSP_XLATE_SET_EVEN_PORT(addr) {
        (
                get <0, addr, PROTO_udp, 10000> from FWX_ALLOC_TABLE_ID to
sr13,
                (

                        sr13 & 1 = 0,

                        (set sr13 (sr13 + 1), RTSP_XLATE_CHECK_PORTS(addr, sr13))
or
```

```
                        (set sr13 (sr13 + 2), RTSP_XLATE_CHECK_PORTS(addr, sr13))
or
                        (set sr13 (sr13 + 2), RTSP_XLATE_CHECK_PORTS(addr, sr13))
or
                        (set sr13 (sr13 + 2), RTSP_XLATE_CHECK_PORTS(addr, sr13))
or
                        (set sr13 (sr13 + 2), RTSP_XLATE_CHECK_PORTS(addr,
sr13)),
                        record <0, addr, PROTO_udp, 10000; sr13> in
FWX_ALLOC_TABLE_ID
                ) or 1

        )
};


/* STR_TO_INT is a macro to convert a string representing an unsigned
// int of max 6 characters long. The result parameter must translate
// into a register. Upon exit, sr2 will point to the char just after the
// string. sr1,2,3 are scratch registers (you can use sr2 for offset
// and/or sr1 for the result. Otherwise use registers sr4 and above).
*/
#define IS_DIGIT      ([sr2:1] >= 0x30, [sr2:1] <= 0x39)
#define ADD_DIGIT ADD_DIGIT_, set sr2 sr2+1
#define STR_TO_INT(offset, result) (
                        \
        set sr2 offset, set sr1 0, set sr3 1,
                \
        IS_DIGIT, ADD_DIGIT, (IS_DIGIT, ADD_DIGIT,
                                \
        IS_DIGIT, ADD_DIGIT, IS_DIGIT, ADD_DIGIT,
                        \
        IS_DIGIT, ADD_DIGIT, IS_DIGIT, ADD_DIGIT) or 1,
                        \
        set result sr1
                                        \
 )


rtsp_tab = dynamic refresh sync expires 60;

/* Search for "client_port=" and continue to "server_port=".
 * sr1-sr4 are scratch registers
 */
#define RTSP_GET_SERVER_PORT(cli_reg,ser_reg,dash_flag) (
        \
```

```
        set cli_reg 0, set ser_reg 0, set dash_flag 0,
        \
        set sr4 call KFUNC_FIND_STR<TCPDATA, RTSP_CL_STR_LEN,
RTSP_CL_STR>,                \
        sr4 != 0, set sr4 sr4 + RTSP_CL_STR_LEN,
                \
        STR_TO_INT(sr4,cli_reg), set sr4 sr2,
                        \
        ([sr4:1] = 0x2d, set dash_flag 1) or 1,
                \
        set sr4 call KFUNC_FIND_STR<sr4, RTSP_SRV_STR_LEN,
RTSP_SRV_STR>,               \
        sr4 != 0, set sr4 sr4 + RTSP_SRV_STR_LEN,
                        \
        STR_TO_INT(sr4,ser_reg)
                                        \
)

/* sr1-sr3 are scratch registers */
#define RTSP_GET_PORT(offset1_reg, port1_reg, offset2_reg, port2_reg) (
        \
        set port2_reg 0,
                                        \
        set offset1_reg call KFUNC_FIND_STR
                                \
                                                <offset1_reg, RTSP_CL_STR_LEN,
RTSP_CL_STR>,                \
        (offset1_reg != 0,
                                        \
                set offset1_reg offset1_reg + RTSP_CL_STR_LEN,
                \
                STR_TO_INT(offset1_reg,port1_reg),
                                \
                ([sr2:1] = 0x2d,
                                        \
                        set sr2 sr2+1, set offset2_reg sr2,
                        \
                        (STR_TO_INT(sr2, port2_reg) or set port2_reg 0)
                \
                ) or 1
                                                \
        ) or set port1_reg 0
                                \
)
```

```
#define RTSP_HANDLE_CL_PORTS (
                                        \
        RTSP_GET_PORT(sr4, sr5, sr6, sr7),
                                \
        (sr5 != 0,
                                        \
                (sr7 != 0,
                                        \
                        get <conn> from FWX_FORW_TABLE_ID to sr9,
                        \
                        RTSP_XLATE_SET_EVEN_PORT(sr9)
                                \
                ) or 1,
                                        \
                NOTSERVER_UDP_PORT(sr5) or reject,
                                \
                RTSP_ANTICIPATE(sr4 - TCPDATA, sr5),
                                \
                (sr7 != 0,
                                        \
                        NOTSERVER_UDP_PORT(sr5) or reject,
                                \
                        RTSP_ANTICIPATE(sr6 - TCPDATA + 1, sr7)
                                \
                ) or 1
                                        \
        )
                                                \
)

/* sr1-sr7 are scratch registers */
#define RTSP_CLIENT_PORTS(port_reg) (
                        \
        set sr4 TCPDATA, RTSP_HANDLE_CL_PORTS, set port_reg sr5,
                \
        (
                                                \
                RTSP_HANDLE_CL_PORTS, RTSP_HANDLE_CL_PORTS,
                                \
                RTSP_HANDLE_CL_PORTS, RTSP_HANDLE_CL_PORTS,
                                \
                RTSP_HANDLE_CL_PORTS
                                        \
        ) or 1
                                        \
 )
```

```
#define RTSP_intercept_client
                                        \
        (
                                                                \
                r_cdir = 1, dport = RTSP, tcp,
                        \
                RTSP_SETUP_HEADER, RTSP_CLIENT_PORTS(sr8),
                                \
                record <conn;sr8> in rtsp_tab,                                  \
                accept_fwz_as_clear(r_ctype)
                        \
        )

#define RTSP_intercept_server
                                        \
        (
                                                                \
       r_cdir = 2, sport = RTSP, tcp,
                \
                RTSP_HEADER, TABLE_NOT_EMPTY(rtsp_tab),
                                        \
                RTSP_GET_SERVER_PORT(sr6,sr7,sr8),
                                \
                get <rconn> from rtsp_tab to sr4,
                        \
                (       get <dst,sr4,src,0,17> from FWX_FORW_TABLE_ID to sr9,
                \
                        set sr10 sr10 & 0xFFFF, sr6 = sr10)
                        \
                or (
                                        \
                        sr6 = sr4
                                        \
                ),
                                                                \
       delete <rconn> from rtsp_tab,
                \
                (
                                                                \
                        (ENTRY_TYPE(r_ctype) = CONN_TCP, set sr5 CONN_UDP)
                        \
                                                                or
                                                        \
                        (set sr5 ENTRY_TYPE(r_ctype))
                                \
```

```
            ),
                                        \
            RTSP_UDP_RECORD(dst,sr4,src,sr7,PROTO_udp,
                        \
                                        0, sr5 | _UDP_ESTABLISHED,
                                \

        TRACKED_TRANS(r_cflags)|RECORD_SRC(0xa6)),                      \
            RTSP_UDP_RECORD(dst,sr4,src,0,PROTO_udp,
                \
                                        DUP_KEY(r_ckey), sr5 |
_UDP_ESTABLISHED,                       \

        TRACKED_TRANS(r_cflags)|RECORD_SRC(0xa6)),                      \
            (ENTRY_TRACKED(r_cflags),                                           \
            record <0,dst,sr4,src,sr7,PROTO_udp;conn@ RTSP_TIMEOUT>
        \

                        in tracked)     \
            or 1,
                                        \
            (
                                                \
                sr8 = 1, set sr4 (sr4 + 1), set sr7 (sr7 + 1),
        \
                RTSP_UDP_RECORD(dst,sr4,src,sr7,PROTO_udp,
                        \
                                                0, sr5 | _UDP_ESTABLISHED,
                                \

        TRACKED_TRANS(r_cflags)|RECORD_SRC(0xa7)),              \
                RTSP_UDP_RECORD(dst,sr4,src,0,PROTO_udp,
                \
                                                DUP_KEY(r_ckey),sr5 |
_UDP_ESTABLISHED,                       \

        TRACKED_TRANS(r_cflags)|RECORD_SRC(0xa7)),              \
                (ENTRY_TRACKED(r_cflags),
\
                record <0,dst,sr4,src,sr7,PROTO_udp;conn@ RTSP_TIMEOUT>
          \

                        in tracked)     \
                or 1
                                        \
```

```
                ) or 1,
                                                \
                accept_fwz_as_clear(r_ctype)
                        \
        )

#define rtsp_prolog  RTSP_intercept_client or RTSP_intercept_server;


#ifndef NO_SQL
```

SQL*Net2

```
#define SQLNET_IS_REDIRECT    (call KFUNC_SQLNET <0>)
#define SQLNET_GET_PORT               (call KFUNC_SQLNET <1>)
#define SQLNET_GET_HOST               (call KFUNC_SQLNET <2>)
#define SQLNET_GET_PROTO      (call KFUNC_SQLNET <3>)
#define SQLNET_MAGIC          0x53514c4e            /* "SQLN" */

/*
 * This only works if the manager and the server are translated through
 * the same DST_STATIC rule (or if the manager and the server are not
 * translated at all).
 */
#ifndef FW_NOXLATION
#define SQLNET_ANTICIPATE(host,port,proto)
                        \
                (call KFUNC_XLATE_ANTICIPATE
                                        \
                        <dst, 0, (host), (port), (proto), BUILD_CONT_REV(0,1),
        \
                                        PENDING_TIMEOUT, 3, 0, 0>)
#else
#define SQLNET_ANTICIPATE(host,port,proto) 1
#endif

//sqlnet_port_tab = { 1521, 1525, 1526 };

#define sqlnet_record
                                \
      record <dst,SQLNET_MAGIC,sr1,sr2,sr3;0,0> in pending,
        \
      (<rconn> in tracked,
                        \
                record <0,dst,SQLNET_MAGIC,sr1,sr2,sr3;rconn
@PENDING_TIMEOUT>               \
```

```
                                        in tracked)       \
            or 1


#define sqlnet_prologue
                                \
        r_cdir = 2, tcp, sport in sqlnet_port_tab,
          \
        SQLNET_IS_REDIRECT,
                                \
                    set sr1 SQLNET_GET_HOST, set sr2 SQLNET_GET_PORT,
                                \
                    set sr3 SQLNET_GET_PROTO,
                                            \
                    direction=1
                                            \
                                or
                                                        \
                    (origsrc=src, origsport=sport)
                                        \
                                or
                                                        \
                    (SQLNET_ANTICIPATE(sr1,sr2,sr3),
                                        \
         SQLNET_IS_REDIRECT,
                                \
                     set sr1 SQLNET_GET_HOST),
                                            \
                    sqlnet_record,
                                            \
                    accept_fwz_as_clear(r_ctype);

#define sqlnet_match
                                        \
                    ((tcp, dport in sqlnet_port_tab)
                                \
                                        or
                                                        \
                    (TABLE_NOT_EMPTY(pending),
                                            \
                     <src,SQLNET_MAGIC,dst,dport,ip_p> in pending,
                                \
                     ACCOUNT_MATCH(SQLNET_MAGIC,0,TCP_TIMEOUT)))

/*
```

```
 * The following is an alternative version of the sqlnet code to use when
 * using sqlnet with resource (as a TCP service).
 */
#define sqlnet_res_record
                                        \
        record <dst,SQLNET_MAGIC,sr1,sr2,sr3;DUP_KEY(r_ckey),r_ctype,r_cflags>\

                                        in pending,     \
        (<rconn> in tracked,
                                \
                        record <0,dst,SQLNET_MAGIC,sr1,sr2,sr3;rconn
@PENDING_TIMEOUT>               \

                                        in tracked)     \
                or 1

#define sqlnet_res_prologue
                                        \
        r_cdir = 2, tcp,
                                \
                        sport in sqlnet_port_tab or sport = auth_services[1521,ip_p],
          \
        SQLNET_IS_REDIRECT,
                                \
                        set sr1 SQLNET_GET_HOST, set sr2 SQLNET_GET_PORT,
                                \
                        set sr3 SQLNET_GET_PROTO,
                                        \
                        direction=1
                                        \
                                or
                                                \
                        (origsrc=src, origsport=sport)
                                        \
                                or
                                                \
                        (SQLNET_ANTICIPATE(sr1,sr2,sr3),
                                        \
         SQLNET_IS_REDIRECT,
                                \
                         set sr1 SQLNET_GET_HOST),
                                        \
                        sqlnet_res_record,
                                        \
                        direction = 0 or SQLNET_ANTICIPATE(0,sr2,sr3),
                                \
```

```
                accept_fwz_as_clear(r_ctype);

#define sqlnet_res_prematch
                                \
                tcp, first, TABLE_NOT_EMPTY(pending),
                        \
                get <src,SQLNET_MAGIC,dst,dport,ip_p> from pending to sr10,
        \
                set sr5 CHANGE_MATCH(sr11,MATCH_BY_PROTOCOL),
                                \
                set sr5 REMOVE_CTRL(sr5, TCP_CTRLS),
                                \
                NOT_TCP_FASTMODE_PORT(sport,0) or reject,
                        \
                record <conn;DUP_KEY(sr10),sr5,IS_ACCEPTED_A|
                        \
                                SPOOF_CACHE_EMPTY|(TRACKED_TRANS(sr12) &
TRACK_UNKNOWN)|         \

        RECORD_SRC(0x84)> in connections,       \
      delete <src,SQLNET_MAGIC,dst,dport,ip_p> from pending,
        \
                (ENTRY_TRACKED(sr12),
                                        \
                ACCOUNT_MATCH(SQLNET_MAGIC,0,TCP_TIMEOUT)) or 1,
                                \
                accept_tcp_noncrypt(sr11,1)

#endif
```

FreeTel

```
#define FREETEL_TIMEOUT 60
#define FREETEL_SERVER 21300
#define FREETEL_CLIENT_FIRST 21301
#define FREETEL_CLIENT_LAST 21305
#define FREETEL_IS_CLIENT(port) \
  ((port) <= FREETEL_CLIENT_LAST , (port) >= FREETEL_CLIENT_FIRST)
#define FREETEL_MAGIC 0x4653544c                /* "FRTL" */

/*
 * intercept the client server connection to refresh the client table
 */
#define freetel_prologue
                                \
```

```
        dport = FREETEL_SERVER, FREETEL_IS_CLIENT(sport), udp,
                \
        TABLE_NOT_EMPTY(pending),
                                        \
        <src,sport,0,FREETEL_MAGIC,ip_p> in pending;

/*
 * check that udp and either server connection (client is in freetel_tab)
 * or client to client connection where the source is in the table.
 */
#define freetel_outgoing
                                \
        (udp,
                                                \
         (dport = FREETEL_SERVER, FREETEL_IS_CLIENT(sport),
                \
          record <src,sport,0,FREETEL_MAGIC,ip_p;0,0@FREETEL_TIMEOUT>
                \

                                in pending)     \
                                or
                                                        \
         (FREETEL_IS_CLIENT(sport), FREETEL_IS_CLIENT(dport),
                \
          <src,sport,0,FREETEL_MAGIC,ip_p> in pending))

/*
 * enable outher side originated calls, destination must be in table
 */
#define freetel_incoming
                                \
        (udp,
                                                \
         FREETEL_IS_CLIENT(sport), FREETEL_IS_CLIENT(dport),
                \
         <dst,dport,0,FREETEL_MAGIC,ip_p> in pending)
```

CoolTalk

Protocol Name: CoolTalk
Date: 14/5/96
Protocol Description:
The protocol is used to pass voice, text and whiteboard drawing between users.
The protocol uses 2 TCP connections:

1. dport=6499 , sport=any - used only for downloading subscribers from a IS411 Server + registering on it. The connection is not a must for cooltalk operation.

2. dport=6500, sport=any - used to establish a connection between 2 cooltalk end users. The connection is used to pass control, chat data and whiteboard data.

The protocol uses UDP packets to pass voice between end users.
dport = 13000, sport = any. Each user will send such packets. So each user has 2 UDP connections, on one of them he receives on port 13000 and on the other sends to port 13000.

Inspect Description:
The 2 TCP connections will be handled as normal TCP connections (defined in the GUI).
cooltalk_intercept intercepts the 6500 TCP connection to cooltalk_datatab.
Any UDP packet with dport=13000 whose <src,dst> or <dst,src> is in cooltalk_datatab is recorded in connections by cooltalk_accept.
This connection will exist at most UDP_TIMEOUT (default =40 secs), however the users may be quiet for more than 40 secs and then talk, for this reason the entries in cooltalk_datatab have a long timeout (1800 sec). cooltalk_rmv looks for the end of the 6500 TCP connection in order to remove it from cooltalk_datatab so it will not stay there after the cooltalk session is over.

```
#define CoolTalkTCP                     6500
#define CoolTalkUDP                             13000
#define IS411TCP                        6499
#define COOLTALK_TIMEOUT        1800
#define COOLTALK_MAGIC                          0x434c544b              /* "CLTK" */


#define cooltalk_find_entry
                        \
                (
                                                        \
                        (get <src,dst> from cooltalk_datatab to sr10, set sr3 1)
        \
                or
                                                        \
                        (get <dst,src> from cooltalk_datatab to sr10, set sr3 2)
        \
                )


#define cooltalk_intercept
                                \
        r_cdir = 1, dport = CoolTalkTCP, tcp,
                \
                <src,dst> in cooltalk_datatab or
                        \
                (
                                                        \
```

```
                        record <src,dst;DUP_KEY(r_ckey),r_ctype,r_cflags>
                \

        in cooltalk_datatab,    \
        (ENTRY_TRACKED(r_cflags),
                        \
                        record
<0,src,COOLTALK_MAGIC,dst,CoolTalkUDP,PROTO_udp;conn                    \

        @PENDING_TIMEOUT> in tracked,           \
                        record
<0,src,CoolTalkUDP,dst,COOLTALK_MAGIC,PROTO_udp;conn                    \

        @PENDING_TIMEOUT> in tracked)           \
                or 1
                                        \
                )


#define cooltalk_rmv                                            \
        (dport = CoolTalkTCP, tcp , tcpdone,
                \
                 delete <src,dst> from cooltalk_datatab
                        \
        )
                                                \
                or                                              \
        (sport = CoolTalkTCP, tcp , tcpdone,                                    \
                 delete <dst,src> from cooltalk_datatab
                        \
        )

/*
 * We set sr1 to UDP_TIMEOUT since otherwise there is a problem
 * of macro argument expansion when compiling
 */

#define cooltalk_accept1
                                \
                dport = CoolTalkUDP, udp,
                                \
                r_cdir = 1 or
                                        \
                (
                                                \
```

```
                        cooltalk_find_entry,
                                \
                        (
                                                        \
                                (ENTRY_TYPE(sr11) = CONN_TCP, set sr2
CONN_UDP) or                    \
                                (set sr2 ENTRY_TYPE(sr11))
                                \
                        ),
#define cooltalk_accept2
                                \
                        (sr3 = 1,
                                                \
                        UDP_RECORD(udpconn,DUP_KEY(sr10),sr2 |
_UDP_ESTABLISHED,               \

                RECORD_SRC(0xa4)),      \
                        UDP_RECORD(conn,0,sr2 |
_UDP_ESTABLISHED,RECORD_SRC(0xa4)))             \
                        or
                                                        \
                        (UDP_RECORD(udprconn,DUP_KEY(sr10),sr2 |
_UDP_ESTABLISHED,               \

                RECORD_SRC(0xa4)),      \
                         UDP_RECORD(rconn,0,sr2 |
_UDP_ESTABLISHED,RECORD_SRC(0xa4))),    \
                        (ENTRY_TRACKED(sr11),
                                \
                        (sr3 = 1,
                                                \

        ACCOUNT_MATCH(COOLTALK_MAGIC,0,UDP_TIMEOUT))
                        \
                        or
                                                        \

        (RACCOUNT_MATCH(COOLTALK_MAGIC,0,UDP_TIMEOUT))) or 1,
                \
                        accept_udp_noncrypt(sr2,sr3)
                        \
                )

#define cooltalk_prolog cooltalk_intercept; cooltalk_rmv;
#define cooltalk_prematch cooltalk_accept1 cooltalk_accept2
```

NetShow

Protocol Name: NetShow
Date: 30/10/96
Protocol Description:
client starts TCP connection on port 1755.
client sends "port" command to server, then server starts UDP on that port to client.
Inspect Description:
netshow_intercept looks for a LVM Connect Funnel Command, that contains the port of UDP connection to follow. It saves the expected UDP connection in pending - to allow it, and in netshow_tab to be able to remove it when the TCP session ends. (We can't count on the UDP connection to last by itself since there may be no UDP packet for a long while, therefore we use the long timeout of pending (3600) and therefore must remove the entry from there by ourselves).
netshow_accept accepts the UDP connections.
netshow_rmv removes the entry for accepted UDP connections when the TCP session ends.

```
#ifndef NO_XLATION
#define NS_ANTICIPATE(port,p_offset,h_offset)
                \
                (call KFUNC_XLATE_ANTICIPATE
                                        \
                 <0, (port), dst, 0, PROTO_udp, BUILD_CONT_REV(0,0),
        \
                                PENDING_TIMEOUT, 4, (p_offset), (h_offset)>)
#else
#define NS_ANTICIPATE(port,p_offset,h_offset) 1
#endif

#define NS_PORT 1755
#define NS_MAGIC 0x4E455453  /* "NETS" */
#define NS_SIGNATURE1 0xCEFA0BB0
#define NS_SIGNATURE2 0x4D4D5320 /* "MMS " */
#define NS_SIGNATURE3 0x02000300
#define NS_VAL  ([sr1:1])
#define NS_VAL2 ([sr1:2, b])
#define NS_SRCH (NS_VAL != 0x5c, (set sr1 (sr1 - 1)))
#define NS_IS_HOST_CMD ([sr1,b] != 0x5C005C00)
#define NS_SRCH2 (NS_IS_HOST_CMD, (set sr1 (sr1 - 2)))
// Make sure the protocol going to be used is UDP
#define NS_USING_UDP  (set sr4 sr1 - 6,
                        \
                                        [sr4,b] = 0x55004400, [sr4+4:2,b] = 0x5000)
#define NS_TIMEOUT 3600
```

```
#define NS_FWD2 set sr1 (sr1 + 2)
#define ASC2DEC set sr2 (sr2*10 + (NS_VAL-0x30))


/* The format of the host/port command is "\\h.h.h.h\UDP\port"
 * (wide-characters).
 * start 10 bytes from end of packet,
 * and search for '\' char (fails if not found).
 * When done, the following registers will contain the following
 * information:
 * sr1 - host offset
 * sr2 - port # of the expected UDP "connection".
 * sr3 - port offset
 */
#define NS_GET_PORT
                                                        \
        set sr1 TCPDATA+ packetlen -41-10,
                                \
        (
                                                        \
                (NS_SRCH, NS_SRCH, NS_SRCH, NS_SRCH, NS_SRCH,
                                \
                 NS_SRCH, NS_SRCH, NS_SRCH, NS_SRCH, NS_SRCH,
                                \
                 NS_SRCH, NS_SRCH, NS_SRCH, NS_SRCH, NS_SRCH,
                                \
                 NS_SRCH, NS_SRCH, NS_SRCH, NS_SRCH, NS_SRCH,
                                \
                 NS_SRCH, NS_SRCH, NS_SRCH, NS_SRCH) or 1
                                \
        ),
                                                        \
        NS_USING_UDP,
                                                        \
        set sr2 0, set sr3 sr1+2,
                                \
        NS_FWD2, ASC2DEC, NS_FWD2, ASC2DEC, NS_FWD2, ASC2DEC,
NS_FWD2, ASC2DEC, \
        NS_FWD2, (NS_VAL != 0, ASC2DEC) or 1,
                                \
        set sr1 sr3 - 20,
                                        \
        (
                                                        \
```

```
                (NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2,
                                        \
                 NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2,
                                        \
                 NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2,
                                        \
                 NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2,
                                        \
                 NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2,
                                        \
                 NS_SRCH2, NS_SRCH2, NS_SRCH2, NS_SRCH2) or 1
                        \
        ),
                                                        \
        set sr1 sr1+4


#define netshow_track
                                        \
                (
                                                        \
                  (
                                                \
                                        ENTRY_TRACKED(r_cflags),
                                                \
                                        record <0,src,sr2,dst,NS_MAGIC,PROTO_udp;conn
@TCP_TIMEOUT>   \

                                        in tracked      \
                        ) or 1
                                                \
                )


#define netshow_intercept
                                        \
        r_cdir = 1, dport = NS_PORT, tcp,
                        \
        [TCPDATA+4,b] = NS_SIGNATURE1, [TCPDATA+12,b] =
NS_SIGNATURE2,                          \
        [TCPDATA+36,b] = NS_SIGNATURE3,
                                        \
        NS_GET_PORT, NOTSERVER_UDP_PORT(sr2) or reject,
                                        \
```

```
        NS_ANTICIPATE(sr2, sr3, sr1),
                                \
        ( (ENTRY_TYPE(r_ctype) = CONN_TCP, set sr3 CONN_UDP ) or
                \
         (set sr3 ENTRY_TYPE(r_ctype))),
                        \
        record <src,sr2,dst,NS_MAGIC,PROTO_udp;DUP_KEY(r_ckey),sr3,r_cflags
        \

        @NS_TIMEOUT> in pending,\
        record <rconn;sr2> in netshow_tab,
                        \
        netshow_track

#define netshow_accept
                                        \
        udp,
                                                \
        TABLE_NOT_EMPTY(pending),
                                        \
        get <dst,dport,src,NS_MAGIC,ip_p> from pending to sr10,
        \
        UDP_RECORD(rconn,0,sr11 | _UDP_ESTABLISHED,
                                \

        TRACKED_TRANS(sr12)|RECORD_SRC(0xaa)), \
        UDP_RECORD(udprconn,DUP_KEY(sr10),sr11 | _UDP_ESTABLISHED,
                        \

        TRACKED_TRANS(sr12)|RECORD_SRC(0xaa)), \
        (ENTRY_TRACKED(sr12),
                                        \
        RACCOUNT_MATCH(NS_MAGIC,0,UDP_TIMEOUT)) or 1,
                                \
        accept_udp_noncrypt(sr11,2)

#define netshow_rmv
                                        \
        dport = NS_PORT, tcp, tcpdone,
                                \
        get <rconn> from netshow_tab to sr2,
                        \
        delete <rconn> from netshow_tab,
                        \
        delete <src,sr2,dst,NS_MAGIC,PROTO_udp> from pending
```

```
#define netshow_prolog netshow_intercept; netshow_rmv;
#define netshow_prematch netshow_accept
```

WinFrame

General discription:
The Winframe protocol -

1. The client makes a TCP connection (SYN), which is accepted due to the rule in the winframe_match macro in the rule base. the connection is recorded in the regular connections table (because of RECORD_CONN), and also added to the wf_connections table, where its sequence number is stored.
2. Now the server responds with a SYN ACK which is intercepted by the prologue and written in the wf_connections table where its sequence number is stored.
3. Next, the client sends an ACK which the prologue lets past it (this is done by identifying the seq of the ACK with the first seq stored in our table.
4. Now the client and server exchange the necessary codes (0x767649434100) The prologue identifies the packets with the code by their seq numbers, and accepts them so that they dont get encrypted. i.e. if encryption is enabled, the codes are sent and received unencrypted. When a code passes, its matching entry is removed from the table. When both entries are absent from the wf_connections table, the rest of the packets in this connection are freely passed along

NOTE: Server browsing by the client will not be possible using the firewall, if the UDP high ports are not permitted by the firewall, i.e. specifying the server in the client must be done by address,not by name.

wf_connections table format:
<conn, server/client direction; first_code_sequence_nuber>

```
/* winframe definitions: */

#define WF_PORT                 1494
#define PACKET_CONTAINS_WF_CODE ( [TCPDATA:1] = 0x7f,        \
                                  [TCPDATA+1:1] = 0x7f,
        \
                                  [TCPDATA+2:1] = 0x49,
        \
                                  [TCPDATA+3:1] = 0x43,
        \
                                  [TCPDATA+4:1] = 0x41,
        \
                                  [TCPDATA+5:1] = 0x00 )
#define second                  (syn, ack)
#define C_TO_S                  1
#define S_TO_C                  2
```

```
#define IS_CODE_SEQ( con_dir )    ( wf_connections[conn,con_dir] = th_seq )

#define WF_CONN_NOT_IN_TABLE        ( not <conn,C_TO_S> in wf_connections,
        \
                                                                not <conn,S_TO_C> in
wf_connections,                 \
                                                                not <rconn,C_TO_S> in
wf_connections,                 \
                                                                not <rconn,S_TO_C> in
wf_connections )

#define DEL_WF_CONN(conn_direction) delete <conn,conn_direction> from
wf_connections

/*
 * winframe_prologue:
 *
 *  For all packets going to the winframe port (1494) -
 *       first one will pass the prologue and be cought in the match rule where
 *       it will enter wf_connections. Second one will be recorded in the
 *  connections also (its sequence number recorded). Third on will pass the
 *  prologue in the row (IS_CODE_SEQ(C_TO_S) because its seq num is recorded.
 *  Fourth and fifth packets which contain the code will be checked (we know
 *  their seq num) after code has been seen, rest of packets will pass because
 *  they are not in the wf_connections table
 */
#define winframe_prologue
                                \
        r_cdir, (sport = WF_PORT or dport = WF_PORT), tcp,
                \
        ( (first)
                                \
          or (WF_CONN_NOT_IN_TABLE)
                                \
          or (second, record <conn,S_TO_C;(th_seq+1)> in wf_connections)
        \
          or (IS_CODE_SEQ(S_TO_C),
                                \
          PACKET_CONTAINS_WF_CODE,
                                \
          direction = 0 or DEL_WF_CONN(S_TO_C), accept_fwz_as_clear(r_ctype))
        \
          or (IS_CODE_SEQ(C_TO_S),
                                \
          PACKET_CONTAINS_WF_CODE,
                                \
```

```
        direction = 1 or DEL_WF_CONN(C_TO_S), accept_fwz_as_clear(r_ctype))
      \
        or (IS_CODE_SEQ(C_TO_S))
                                        \
        or (drop) );

#define winframe_match      ( tcp, dport = WF_PORT,
                \
                                                record <conn,C_TO_S;(th_seq+1)> in
wf_connections )
```

BACKWEB

Client initiates a connection to the server, sends a challenge and asks to connect. The server then responds with its challenge. From then on both client and server exchange data packets with the other's challenge, for extra identification.

```
#define BW_dport            370
#define BW_sport            371
#define BW_MAGIC            0x42574542    /* "BWEB" */
#define BW_TIMEOUT                60
#define BW_type                   ([UDPDATA:1] & 0x1f)
#define BWp_challenge             [(UDPDATA+4):4]
#define BWs_challenge             [(UDPDATA+8):4]

#define BW_CONNECT                (BW_type = 0x1)
#define BW_CONNECTION_GRANTED   (BW_type = 0x2)
#define BW_DATA                   (BW_type = 0x3)
#define BW_NOT_CONNECTED_ERROR (BW_type = 0x4)



#define backweb_match
                                        \
                (sport=BW_sport, dport=BW_dport, udp, BW_CONNECT,
                \
                record
<src,sport,dst,BW_MAGIC,ip_p;0,0,BWp_challenge,0@BW_TIMEOUT> \

                            in pending)

#define bw_conn_granted (
                            \
                BW_CONNECTION_GRANTED,
                                        \
```

```
                get <dst,dport,src,BW_MAGIC,ip_p> from pending to sr1,
        \
                record
<dst,dport,src,BW_MAGIC,ip_p;0,0,sr3,BWs_challenge@BW_TIMEOUT> \

                                in pending      \
)

#define bw_c_data (
                                        \
                BW_DATA,
                                        \
                get <src,sport,dst,BW_MAGIC,ip_p> from pending to sr1,
        \
                sr4!=BWp_challenge
                                        \
)

#define bw_s_data_or_error (
                        \
                (BW_DATA or BW_NOT_CONNECTED_ERROR),
                                        \
                get <dst,dport,src,BW_MAGIC,ip_p> from pending to sr1,
        \
                sr3!=BWp_challenge
                                        \
)

#define backweb_conn {          udp,
                                \
                (sport=BW_sport, dport=BW_dport,
                        \
                        ((bw_c_data, drop)
                                        \
                        or
                                                \
                        (BW_CONNECT, <src,sport,dst,BW_MAGIC,ip_p> in pending,
                \
                        record <src,sport,dst,BW_MAGIC,ip_p;0,0,BWp_challenge,0
        \

        @BW_TIMEOUT> in pending,        \
                        accept)))
                                        \
                        or
                                                \
```

```
                    (dport=BW_sport,
                                \
                    ((bw_conn_granted, accept)
                    \
                    or
                                        \
                    (bw_s_data_or_error, drop)));
                    \
}
```

IIOP

```
#define IIOP_MAGIC1                     0x49494f31    /* "IIO1" */
#define IIOP_MAGIC2                     0x49494f32    /* "IIO2" */

/* Comment the following line to disable IIOP name service */

#define ENABLE_IIOP_NAMESERVICE

/* Comment the following line to disable IIOP orbix daemon mapping */

#define ENABLE_IIOP_ORBIXD

/* Comment the following line to disable IIOP callbacks */

#define ENABLE_IIOP_CALLBACKS


#define IIOP_MAGIC_ORBIXD       0x01
#define IIOP_MAGIC_RESOLVE      0x02

#define IIOP_HEADER                     ([TCPDATA,b] = 0x47494f50) /*
"GIOP" */
#define IIOP_MAJOR                      [TCPDATA+4:1]
#define IIOP_MINOR                      [TCPDATA+5:1]
#define IIOP_FLAGS              [TCPDATA+6:1]
#define IIOP_MSG_TYPE           [TCPDATA+7:1]
#define IIOP_RESP_EXPECTED      ([TCPDATA+20:1] = 1)
#define IIOP_REQUEST_ID_B       [TCPDATA+16,b]
#define IIOP_REQUEST_ID_L       [TCPDATA+16,l]
#define IIOP_ORBIXD_PORTLEN_B   [TCPDATA+24,b]
#define IIOP_ORBIXD_PORTLEN_L   [TCPDATA+24,l]
```

```
#define IIOP_REPLY_STATUS_B          [TCPDATA+20,b]
#define IIOP_REPLY_STATUS_L          [TCPDATA+20,l]
#define IIOP_LOCATE_STATUS_B         [TCPDATA+16,b]
#define IIOP_LOCATE_STATUS_L         [TCPDATA+16,l]

#define IIOP_REQUEST                 (IIOP_MSG_TYPE = 0)
#define IIOP_REPLY                   (IIOP_MSG_TYPE = 1)
#define IIOP_LOCATEREPLY             (IIOP_MSG_TYPE = 4)
#define IIOP_BYTEORDER               (IIOP_FLAGS & 0x01)
#define IIOP_BIG_ENDIAN              (IIOP_BYTEORDER = 0)
#define IIOP_LITTLE_ENDIAN           (IIOP_BYTEORDER = 1)

#define IIOP_ALIGN2(x)               (((x)+1) & 0xfffffffe)
#define IIOP_ALIGN4(x)               (((x)+3) & 0xfffffffc)

#define IIOP_LOCATE_FORWARD
                                     \
        (
                                                 \
                (IIOP_BIG_ENDIAN, IIOP_REPLY_STATUS_B = 3)
                                 \
                  or
                                     \
                (IIOP_LITTLE_ENDIAN, IIOP_REPLY_STATUS_L = 3)
                         \
        )

#define IIOP_OBJECT_FORWARD
                                   \
        (
                                                 \
                (IIOP_BIG_ENDIAN, IIOP_LOCATE_STATUS_B = 2)
                                 \
                  or
                                     \
                (IIOP_LITTLE_ENDIAN, IIOP_LOCATE_STATUS_L = 2)
                                 \
        )

#define IIOP_OBJECT_KEY_PORT(reg,offset)
                         \
        (
                                                 \
                (
                                                 \
```

```
                IIOP_BIG_ENDIAN,
                                \
                set sr2 ((offset) + 4 + IIOP_ALIGN4(([TCPDATA+(offset),b]))),
    \
                set sr2 (sr2 + 16),
                                \
                set sr2 (sr2 + 4 + IIOP_ALIGN2(([(TCPDATA+sr2),b]))),
    \
                set reg [(TCPDATA+sr2):2,b]
                                \
            ) or (
                                        \
                IIOP_LITTLE_ENDIAN,
                                        \
                set sr2 ((offset) + 4 + IIOP_ALIGN4(([TCPDATA+(offset),l]))),
    \
                set sr2 (sr2 + 16),
                                \
                set sr2 (sr2 + 4 + IIOP_ALIGN2(([(TCPDATA+sr2),l]))),
    \
                set reg [(TCPDATA+sr2):2,l]
                        \
            )
                                                \
    )

#define IIOP_ORBIXD_PORT_5
                                \
      ((([(TCPDATA+28):1] - 0x30) * 10000) +
                \
       (([(TCPDATA+29):1] - 0x30) * 1000) +
                \
       (([(TCPDATA+30):1] - 0x30) * 100) +
                \
       (([(TCPDATA+31):1] - 0x30) * 10) +
                \
       (([(TCPDATA+32):1] - 0x30)))

#define IIOP_ORBIXD_PORT_4
                                \
      ((([(TCPDATA+28):1] - 0x30) * 1000) +
                \
       (([(TCPDATA+29):1] - 0x30) * 100) +
                \
       (([(TCPDATA+30):1] - 0x30) * 10) +
                \
```

```
        (([(TCPDATA+31):1] - 0x30)))

#define IIOP_ORBIXD_PORT_3
                                        \
        ((([(TCPDATA+28):1] - 0x30) * 100) +
                        \
        (([(TCPDATA+29):1] - 0x30) * 10) +
                        \
        (([(TCPDATA+30):1] - 0x30)))

#define IIOP_ORBIXD_PORT(reg)
                                \
        (
                                                        \
                (
                                                        \
                        IIOP_BIG_ENDIAN,
                                        \
                        (IIOP_ORBIXD_PORTLEN_B = 6, set reg
IIOP_ORBIXD_PORT_5)                     \
                                        or
                                                        \
                        (IIOP_ORBIXD_PORTLEN_B = 5,  set reg
IIOP_ORBIXD_PORT_4)                     \
                                        or
                                                        \
                        (IIOP_ORBIXD_PORTLEN_B = 4, set reg
IIOP_ORBIXD_PORT_3)                     \
                ) or (
                                        \
                        IIOP_LITTLE_ENDIAN,
                                        \
                        (IIOP_ORBIXD_PORTLEN_L = 6, set reg
IIOP_ORBIXD_PORT_5)                     \
                                        or
                                                        \
                        (IIOP_ORBIXD_PORTLEN_L = 5,  set reg
IIOP_ORBIXD_PORT_4)                     \
                                        or
                                                        \
                        (IIOP_ORBIXD_PORTLEN_L = 4,  set reg
IIOP_ORBIXD_PORT_3)                     \
                )
                                                        \
        )
```

```
#define IIOP_ORBIXD_GETDLEN          0x0f
#define IIOP_ORBIXD_GETD1            0x67657449          /* "getI" */
#define IIOP_ORBIXD_GETD2            0x494f5044          /* "IOPD" */
#define IIOP_ORBIXD_GETD3            0x65746169          /* "etai" */
#define IIOP_ORBIXD_GETD4            0x6c73              /* "ls"   */

#ifdef  ENABLE_IIOP_ORBIXD
#define IIOP_ORBIXD_SCAN
                                        \
        (
                                                \
                (
                                                \
                        IIOP_BIG_ENDIAN,
                                        \
                        set sr2 (24 + 4 + IIOP_ALIGN4(([TCPDATA+24,b]))),
                \
                        [TCPDATA+sr2,b] = IIOP_ORBIXD_GETDLEN
                                \
                ) or (
                                        \
                        IIOP_LITTLE_ENDIAN,
                                        \
                        set sr2 (24 + 4 + IIOP_ALIGN4(([TCPDATA+24,l]))),
                \
                        [TCPDATA+sr2,l] = IIOP_ORBIXD_GETDLEN
                                \
                ),
                                                \
                [TCPDATA+sr2+4,b] = IIOP_ORBIXD_GETD1,
                                \
                [TCPDATA+sr2+8,b] = IIOP_ORBIXD_GETD2,
                                \
                [TCPDATA+sr2+12,b] = IIOP_ORBIXD_GETD3,
                                \
                [TCPDATA+sr2+16:2,b] = IIOP_ORBIXD_GETD4
                                \
        )
#else
#define IIOP_ORBIXD_SCAN (0)
#endif

#define IIOP_RESOLVELEN              0x08
#define IIOP_RESOLVE1        0x7265736f          /* "reso" */
#define IIOP_RESOLVE2        0x6c766500          /* "lve"  */
```

```
#ifdef  ENABLE_IIOP_NAMESERVICE
#define IIOP_RESOLVE_SCAN
                                        \
        (
                                                        \
                (
                                                        \
                        IIOP_BIG_ENDIAN,
                                        \
                        set sr2 (24 + 4 + IIOP_ALIGN4(([TCPDATA+24,b]))),
                \
                        [TCPDATA+sr2,b] = IIOP_RESOLVELEN
                                \
                ) or (
                                                \
                        IIOP_LITTLE_ENDIAN,
                                                \
                        set sr2 (24 + 4 + IIOP_ALIGN4(([TCPDATA+24,l]))),
                \
                        [TCPDATA+sr2,l] = IIOP_RESOLVELEN
                                \
                ),
                                                        \
                [TCPDATA+sr2+4,b] = IIOP_RESOLVE1,
                                \
                [TCPDATA+sr2+8,b] = IIOP_RESOLVE2
                                \
        )
#else
#define IIOP_RESOLVE_SCAN (0)
#endif


#define IIOP_REGCBACKLEN        0x0f
#define IIOP_REGCBACK1          0x52656769          /* "Regi" */
#define IIOP_REGCBACK2          0x73746572          /* "ster"  */
#define IIOP_REGCBACK3          0x436c6965          /* "Clie"  */
#define IIOP_REGCBACK4          0x6e740000          /* "nt"  */
#define IIOP_IDLCBACKLEN        0x11
#define IIOP_IDLCBACK1          0x49444c3a          /* "IDL:" */
#define IIOP_IDLCBACK2          0x43616c6c          /* "Call"  */
#define IIOP_IDLCBACK3          0x4261636b          /* "Back"  */

#ifdef  ENABLE_IIOP_CALLBACKS
#define IIOP_REGISTER_CALLBACK_SCAN(reg)
                                \
```

```
(
                                                    \
        (
                                                    \
                IIOP_BIG_ENDIAN,
                                \
                set reg (24 + 4 + IIOP_ALIGN4(([TCPDATA+24,b]))),
        \
                [TCPDATA+reg,b] = IIOP_REGCBACKLEN
                                \
        ) or (
                                        \
                IIOP_LITTLE_ENDIAN,
                                        \
                set reg (24 + 4 + IIOP_ALIGN4(([TCPDATA+24,l]))),
        \
                [TCPDATA+reg,l] = IIOP_REGCBACKLEN
                                \
        ),
                                                    \
        [TCPDATA+reg+4,b] = IIOP_REGCBACK1,
                                \
        [TCPDATA+reg+8,b] = IIOP_REGCBACK2,
                                \
        [TCPDATA+reg+12,b] = IIOP_REGCBACK3,
                        \
        [TCPDATA+reg+16,b] = IIOP_REGCBACK4,
                        \
        (
                                                    \
                IIOP_BIG_ENDIAN,
                                \
                set reg (reg + 4 + IIOP_ALIGN4(([(TCPDATA+reg),b]))),
\
                set reg (reg + 4 + IIOP_ALIGN4(([(TCPDATA+reg),b]))),
\
                [TCPDATA+reg,b] = IIOP_IDLCBACKLEN
                                \
        ) or (
                                        \
                IIOP_LITTLE_ENDIAN,
                                        \
                set reg (reg + 4 + IIOP_ALIGN4(([(TCPDATA+reg),l]))),
\
                set reg (reg + 4 + IIOP_ALIGN4(([(TCPDATA+reg),l]))),
\
```

```
                        [TCPDATA+reg,l] = IIOP_IDLCBACKLEN
                                                \
                ),
                                                                \
                [TCPDATA+reg+4,b] = IIOP_IDLCBACK1,
                                                \
                [TCPDATA+reg+8,b] = IIOP_IDLCBACK2,
                                                \
                [TCPDATA+reg+12,b] = IIOP_IDLCBACK3
                                                \
        )
#else
#define IIOP_REGISTER_CALLBACK_SCAN (0)
#endif


#ifndef NO_XLATION
#define IIOP_ANTICIPATE(port)
                                \
        (call KFUNC_XLATE_ANTICIPATE
                                                \
                        <dst, 0, 0, (port), PROTO_tcp, BUILD_CONT_REV(0,1),
                \
                                PENDING_TIMEOUT, 0, 0, 0>)
#define IIOP_ANTICIPATE_CALLBACK(port,port_offset)
                        \
        (call KFUNC_XLATE_ANTICIPATE
                                                \
                        <0, (port), dst, 0, PROTO_tcp, BUILD_CONT_REV(0,0),
                \
                                PENDING_TIMEOUT, 0, (port_offset), 0>)
#else
#define IIOP_ANTICIPATE(port) 1
#define IIOP_ANTICIPATE_CALLBACK(port,port_offset) 1
#endif


#define iiop_intercept_request
                                \
        (
                                                                \
                r_cdir = 1, tcp, dport in iiop_port_tab,
                \
                IIOP_REQUEST, IIOP_RESP_EXPECTED,
                                                \
```

```
(
                                                \
        IIOP_ORBIXD_SCAN,
                                \
        (
                                                \
                IIOP_BIG_ENDIAN,
                        \
                record <conn; IIOP_MAGIC_ORBIXD,
IIOP_REQUEST_ID_B>              \

    in iiop_requests        \
        ) or (
                                \
                IIOP_LITTLE_ENDIAN,
                                \
                record <conn; IIOP_MAGIC_ORBIXD,
IIOP_REQUEST_ID_L>              \

    in iiop_requests        \
        )
                                                \
    ) or (
                                \
        IIOP_RESOLVE_SCAN,
                                \
        (
                                                \
                IIOP_BIG_ENDIAN,
                        \
                record <conn; IIOP_MAGIC_RESOLVE,
IIOP_REQUEST_ID_B>              \

    in iiop_requests        \
        ) or (
                                \
                IIOP_LITTLE_ENDIAN,
                                \
                record <conn; IIOP_MAGIC_RESOLVE,
IIOP_REQUEST_ID_L>              \

    in iiop_requests        \
        )
                                                \
    )
                                                \
```

```
        )


#define iiop_intercept_reply
                        \
        (
                                                        \
                r_cdir = 2, tcp, sport in iiop_port_tab,
                \
                (
                                                        \
                        IIOP_REPLY,
                                                \
                        (
                                                        \
                                IIOP_LOCATE_FORWARD,
                                                \
                                IIOP_OBJECT_KEY_PORT(sr1,24)
                                \
                        ) or (
                                                \
                                get <rconn> from iiop_requests to sr3,
                        \
                                (
                                                        \
                                        IIOP_BIG_ENDIAN, sr4 = IIOP_REQUEST_ID_B
                        \
                                ) or (
                                                \
                                        IIOP_LITTLE_ENDIAN, sr4 =
IIOP_REQUEST_ID_L                                       \
                                ),
                                                        \
                                (
                                                        \
                                        sr3 = IIOP_MAGIC_ORBIXD,
                                        \
                                        IIOP_ORBIXD_PORT(sr1)
                                        \
                                ) or (
                                                \
                                        sr3 = IIOP_MAGIC_RESOLVE,
                                        \
                                        IIOP_OBJECT_KEY_PORT(sr1,24)
                                \
```

```
                        ),
                                                \
                        delete <rconn> from iiop_requests
                \
                )
                                                \
        ) /*or (
                                \
                IIOP_LOCATEREPLY,
                                        \
                (
                                                \
                        IIOP_OBJECT_FORWARD,
                                \
                        IIOP_OBJECT_KEY_PORT(sr1)
                                \
                )
                                                \
        ) */,
                                        \
        IIOP_ANTICIPATE(sr1),
                                \
        record <dst,IIOP_MAGIC1,src,sr1,ip_p;DUP_KEY(r_ckey),r_ctype,
    \

    r_cflags> in pending,  \
        record <src,sr1> in iiop_servers,
                \
  (ENTRY_TRACKED(r_cflags),
                        \
        record <0,dst,IIOP_MAGIC1,src,sr1,ip_p;rconn
@PENDING_TIMEOUT>               \

                        in tracked)     \
        or 1,
                                        \
        accept_fwz_as_clear(r_ctype)
                \
    )


#define iiop_prematch
                                \
    (
                                                \
```

```
tcp, first, TABLE_NOT_EMPTY(pending),
        \
(
                                        \
        get <src,IIOP_MAGIC1,dst,dport,ip_p> from pending to sr10,
\
        set sr5 CHANGE_MATCH(sr11,MATCH_BY_PROTOCOL),
                \
        set sr5 REMOVE_CTRL(sr5, TCP_CTRLS),
                \
        NOT_TCP_FASTMODE_PORT(sport,0) or reject,
        \
        record
<conn;DUP_KEY(sr10),sr5,IS_ACCEPTED_A|MORE_INSPECTION|    \

SPOOF_CACHE_EMPTY|TRACKED_TRANS(sr12)|RECORD_SRC(0x66)>
\

        in connections,         \
    delete <src,IIOP_MAGIC1,dst,dport,ip_p> from pending,
\
        (ENTRY_TRACKED(sr12),
                        \
        ACCOUNT_MATCH(IIOP_MAGIC1,0,TCP_TIMEOUT)) or 1,
                \
        accept_tcp_noncrypt(sr11,1)
                \
) or (
                                \
        get <src,IIOP_MAGIC2,dst,dport,ip_p> from pending to sr10,
\
        set sr5 CHANGE_MATCH(sr11,MATCH_BY_PROTOCOL),
                \
        set sr5 REMOVE_CTRL(sr5, TCP_CTRLS),
                \
        NOT_TCP_FASTMODE_PORT(sport,0) or reject,
        \
        record <rconn;DUP_KEY(sr10),sr5,IS_ACCEPTED_B|
        \

SPOOF_CACHE_EMPTY|TRACKED_TRANS(sr12)|RECORD_SRC(0x67)>
\

        in connections,         \
    delete <src,IIOP_MAGIC2,dst,dport,ip_p> from pending,
\
```

```
                (ENTRY_TRACKED(sr12),
                                \
                RACCOUNT_MATCH(IIOP_MAGIC2,0,TCP_TIMEOUT)) or 1,
                \
                accept_tcp_noncrypt(sr11,2)
                        \
            )
                                                \
        )

#define iiop_intercept_register_callback
            \
        (
                                                \
            r_cdir = 1, tcp, <dst,dport> in iiop_servers,
        \
            IIOP_REQUEST,
                                        \
            IIOP_REGISTER_CALLBACK_SCAN(sr3),
                                \
            IIOP_OBJECT_KEY_PORT(sr1,sr3),
                                \
            NOTSERVER_TCP_PORT(sr1) or reject,
                        \
            IIOP_ANTICIPATE_CALLBACK(sr1,TCPDATA+sr2),
                        \
            record <dst,IIOP_MAGIC2,src,sr1,ip_p;DUP_KEY(r_ckey),r_ctype,
        \

        r_cflags> in pending,  \
      (ENTRY_TRACKED(r_cflags),
                        \
            record <0,dst,IIOP_MAGIC2,src,sr1,ip_p;rconn
@PENDING_TIMEOUT>           \

                        in tracked)     \
            or 1,
                                        \
            accept_fwz_as_clear(r_ctype)
                \
        )


deffunc iiop_prolog() {
        (
```

```
                IIOP_HEADER,

                iiop_intercept_request

                        or

                iiop_intercept_reply

                        or

                iiop_intercept_register_callback

        )
};

#define iiop_port(port)
                                        \
        (
                                                        \
                tcp, dport = (port), record <(port)> in iiop_port_tab
        \
        )
```

NetBios

```
#define NBNAME                          137
#define NBDATAGRAM                      138

/* NetBios Datagram Service */

#define NBDS_DATAGRAM_TYPE          [UDPDATA:1]
#define NBDS_DIRECT_UNIQUE   (NBDS_DATAGRAM_TYPE = 0x10)
#define NBDS_DIRECT_GROUP    (NBDS_DATAGRAM_TYPE = 0x11)

/* NetBios Name Service */

#define NBNS_XID                [UDPDATA:2,b]
#define NBNS_RESP_FLAG          ((([(UDPDATA+2):1]) >> 7) & 0x01)
#define NBNS_REQUEST            (NBNS_RESP_FLAG = 0)
#define NBNS_RESPONSE           (NBNS_RESP_FLAG = 1)

#define NBNS_OPCODE                     ((([(UDPDATA+2):1]) >> 3) & 0x0f)
#define NBNS_QUERY                      (NBNS_OPCODE = 0x00)
```

```
#define NBNS_REGISTRATION     ((NBNS_OPCODE = 0x05) or (NBNS_OPCODE =
0x0f))
#define NBNS_RELEASE          (NBNS_OPCODE = 0x06)
#define NBNS_WACK                     (NBNS_OPCODE = 0x07)
#define NBNS_REFRESH          (NBNS_OPCODE = 0x08)

#define NBNS_RCODE                    (([(UDPDATA+3):1]) & 0x0f)
#define NBNS_SUCCESS          (NBNS_RCODE = 0)

#define NBDS_SRC_NAME_LEN     [UDPDATA + 14:1]
#define NBDS_DST_NAME_LEN     [UDPDATA + 14 + NBDS_SRC_NAME_LEN +
2:1]
#define NBDS_SMB_DATA         \
        (UDPDATA +14 + NBDS_SRC_NAME_LEN + NBDS_DST_NAME_LEN + 4)

/* SMB */
#define SMB_DATA_OFFSET               [NBDS_SMB_DATA + 57:2,l]

#define NETLOGON_OPCODE               [NBDS_SMB_DATA +
SMB_DATA_OFFSET:2,l]
#define LOGON_REQUEST         (NETLOGON_OPCODE = 0)
#define LOGON_RESPONSE                (NETLOGON_OPCODE = 6)

#define NBT_XLATE 0
#define NBT_UNXLATE 1

#define FWX_SRC 0
#define FWX_DST 1

#ifndef NO_XLATION
#define NBT_ANTICIPATE(host_offset,method)
                        \
        (call KFUNC_XLATE_ANTICIPATE
                                \
                <conn, BUILD_CONT_REV((method),0), 63, 0, 0, (host_offset) >)
#define NBT_PREDICT(offset)
                                \
        (call KFUNC_XLATE_PREDICT <offset,FWX_SRC,NBT_XLATE>)
#else
#define NBT_ANTICIPATE(host_offset,method)    (1)
#define NBT_PREDICT(offset)                           (1)
#endif


#define nbdatagram_code
                                \
```

```
        (
                                                        \
                udp, dport = NBDATAGRAM,
                                        \
                NBDS_DIRECT_UNIQUE or NBDS_DIRECT_GROUP,
                                \
                (r_cdir != 2, NBT_ANTICIPATE(UDPDATA+4,1))
                                \
                  or
                                        \
                (r_cdir = 2, NBT_ANTICIPATE(UDPDATA+4,3))
                        \
        )

#define nbname_code
                                        \
        (
                                                        \
                udp,
                                        \
                (
                                                        \
                        dport = NBNAME, NBNS_REQUEST,
                                        \
                        NBNS_REGISTRATION or NBNS_RELEASE or
NBNS_REFRESH,                           \
                        [(UDPDATA+64),b] = src,
                                        \
                        NBT_ANTICIPATE(UDPDATA+64,1)
                                        \
                ) or (
                                        \
                        sport = NBNAME, NBNS_RESPONSE,
                                        \
                        (
                                                        \
                                NBNS_REGISTRATION or NBNS_RELEASE or
NBNS_REFRESH,                           \
                                NBT_ANTICIPATE(UDPDATA+58,4)
                                        \
                        ) or (
                                        \
                                NBNS_QUERY, NBNS_SUCCESS, direction = 0,
                        \
                                NBT_PREDICT(UDPDATA+58)
                                        \
```

```
                )
                                                \
            )
                                                \
        )

#define netbios_code nbdatagram_code or nbname_code;
```

Novell NetWare Core Protocol (NCP) NAT Support

```
#define NETWARE_XLATE           0
#define NETWARE_FWX_SRC                 0

#define NETWARE_NCP_PORT                524

#define NCP_REPLY_HEADER_SIZE   16
#define NCP_REQ_TYPE            [TCPDATA+16:2,b]
#define NCP_REQUEST                     0x2222
#define NCP_REP_TYPE            [TCPDATA+8:2,b]
#define NCP_REPLY               0x3333
#define NCP_FUNC_CODE           [TCPDATA+22,b]
#define NCP_REQ_KEY             [TCPDATA+18,b]
#define NCP_REP_KEY                     [TCPDATA+10,b]
#define NCP_NDS_MASK            0xffff0000
#define FUNC_NDS                0x68020000
#define FUNC_GET_INET_ADDR      0x1700051a
#define FUNC_ENUM_NET_ADDR              0x7b000511

#define NDS_VERB                [TCPDATA+40,l]
#define NDS_REPLY_HEADER_SIZE   12
#define NDS_REPLY_END           (TCPDATA + NCP_REPLY_HEADER_SIZE +
NDS_REPLY_HEADER_SIZE) /* Skip NCP and NDS headers */

#define DSV_RESOLVE_NAME_VERB   1
#define GET_SERVER_ADDR_VERB    53
#define DSV_READ_VERB           3
#define DSV_LINK_REPLICA_VERB   30

#define NETADDR_SYNTAX                  12

#define nds                             (sr1 & NCP_NDS_MASK = FUNC_NDS)

#define dsv_resolve_name        (sr2 = DSV_RESOLVE_NAME_VERB)
```

```
#define get_server_addr        (sr2 = GET_SERVER_ADDR_VERB)
#define dsv_read                       (sr2 = DSV_READ_VERB)
#define dsv_link_replica       (sr2 = DSV_LINK_REPLICA_VERB)

#define ENUM_NET_ADDR_OFF   28
#define enum_net_addr                  (sr1 = FUNC_ENUM_NET_ADDR)

#define NETWARE_PREDICT(offset)
                                       \
        (
                                                   \
                call
KFUNC_XLATE_PREDICT<(offset),NETWARE_FWX_SRC,NETWARE_XLATE>
        \
        )

#define PADDING(value)                 (((value) + 3) & 0xfffffffc)

#define JUMP_FIELD(reg)                (set reg ((reg) + 4 + [(reg),l])), (set reg
PADDING(reg))
/* [reg,l] is the field length, showing how much we need to skip */

ncp_table = dynamic refresh expires 30;

/*******************************************************
            Request Macros
*******************************************************/


#define NCP_PUT_IN_TABLE record <conn, NCP_REQ_KEY; sr1, sr2> in ncp_table

#define ncp_delete_from_table delete <rconn, NCP_REP_KEY> from ncp_table

#define handle_nds_req
                                       \
        (
                                                   \
                nds, set sr1 FUNC_NDS,
                                       \
                set sr2 NDS_VERB,
                                       \
                get_server_addr or dsv_resolve_na me or dsv_read or dsv_link_replica,\
                NCP_PUT_IN_TABLE
                                               \
        )
```

```
#define handle_stat_req (enum_net_addr /*or enum_conn_info or get_server_info or \

get_routers_info or get_known_info or get_sources_info or lan_config_info*/,
NCP_PUT_IN_TABLE)

#define get_inet_addr_req
                                \
        (
                                                        \
                set sr1 NCP_FUNC_CODE, sr1 = FUNC_GET_INET_ADDR,
NCP_PUT_IN_TABLE        \
        )

#define netware_req_match
                                \
        (
                                                        \
                tcp, dport = NETWARE_NCP_PORT, NCP_REQ_TYPE =
NCP_REQUEST,                    \
                set sr1 NCP_FUNC_CODE,
                                \
                handle_nds_req or handle_stat_req or get_inet_addr_req
        \
        )


/**********************************************
            Reply Macros
**********************************************/

/*
 * The following macro looks for the IP address in the address list.
 * The list is of the following structure:
 * 4 bytes - transport type
 * 4 bytes - address length
 * and then the address itself.
 * the code parses this structure, finding the IP address if this is IP,
 * or skipping the address if it is not IP.
 */
#define ADDR_LOOP1
                                                \
        (
                                                        \
                packetlen > sr3, sr4 > 0, set sr4 sr4 - 1,
                \
```

```
(
                                                    \
        [sr3,l] = 8 or [sr3,l] = 9 or [sr3,l] = 6 or [sr3, l] = 5,          \
        [sr3+4,l] = 6, set sr3 sr3 + 10,
\
        NETWARE_PREDICT(sr3) or 1, set sr3 sr3 + 4
        \
) or (
                                        \
        set sr3 (sr3 + 4 + 4 + [sr3 + 4,l])
\
),
                                                    \
enum_net_addr or set sr3 PADDING(sr3)
        \
)

#define ADDR_LOOP2
                                        \
(
                                                    \
packetlen > sr3, sr4 > 0, set sr4 sr4 - 1,
\
(
                                                    \
        [sr3,l] = 14, [sr3+4,l] = 8 or [sr3+4,l] = 9 or [sr3+4,l] = 6,   \
        [sr3+8,l] = 6, set sr3 sr3 + 14,
\
        NETWARE_PREDICT(sr3), set sr3 sr3 + 4
        \
) or (
                                        \
        JUMP_FIELD(sr3)
                                        \
),
                                                    \
set sr3 PADDING(sr3)
                                \
)

#define REFERRAL_LOOP
                                \
(
                                                    \
packetlen > sr3, sr5 > 0,
                        \
```

```
                set sr5 sr5 -1, set sr4 [sr3,l], set sr3 sr3+4,
        \
                ADDR_LOOP1, ADDR_LOOP1
                                        \
        )

#define LOCAL_ENTRY 1
#define REMOTE_ENTRY 2
#define NO_SUCH_ENTRY 0
#define ALIAS_ENTRY 3
#define REFERRAL_INFO 4
#define NAME_AND_REFERRALS 6

#define local_entry_code
                        \
        (
                                                \
                set sr3 sr3+8, set sr4 [sr3,l], set sr3 sr3+4,
        \
                ADDR_LOOP1, ADDR_LOOP1, ADDR_LOOP1
                                \
        )

#define name_and_referrals_code
                        \
        (
                                                \
                set sr3 sr3+12, set sr5 [sr3,l], set sr3 sr3+4,
        \
                REFERRAL_LOOP, REFERRAL_LOOP,
                                \
                REFERRAL_LOOP
                                        \
        )

#define handle_dsv_resolve
                        \
        (
                                                \
                set sr6 [sr3,l],
                        \
                (
                                                \
                        sr6 = LOCAL_ENTRY, local_entry_code or 1
                        \
```

```
                ) or (
                                                \
                        sr6 = NAME_AND_REFERRALS, name_and_referrals_code or 1
                \
                )
                                                        \
        )

#define handle_get_server_addr
                                \
        (
                                                        \
                JUMP_FIELD(sr3), set sr4 [sr3,l],
                        \
                /* sr4 now includes the number of addresses */
                \
                set sr3 sr3 + 4,
                                \
                ADDR_LOOP1, ADDR_LOOP1, ADDR_LOOP1
                                        \
        )

#define handle_dsv_read
                                        \
        (
                                                        \
                set sr3 sr3 + 12, [sr3,l] = NETADDR_SYNTAX, set sr3 sr3 + 4,
        \
                JUMP_FIELD(sr3), set sr4 [sr3,l], set sr3 sr3 + 4,
        \
                ADDR_LOOP2, ADDR_LOOP2, ADDR_LOOP2
                                        \
        )

#define handle_dsv_link_replica
                                \
        (
                                                        \
                0 /* TBD */
                                                \
        )

#define handle_nds_reply
                                \
        (
                                                        \
```

```
                set sr3 NDS_REPLY_END,
                                        \
                (
                                                        \
                        dsv_resolve_name, handle_dsv_resolve or 1
                \
                ) or (
                                                \
                        get_server_addr, handle_get_server_addr or 1
                \
                ) or (
                                                \
                        dsv_read, handle_dsv_read or 1
                                \
                ) or (
                                                \
                        dsv_link_replica, handle_dsv_link_replica
                \
                )
                                                        \
        )

#define handle_enum_net_addr
                                \
        (
                                                        \
                set sr3 TCPDATA + NCP_REPLY_HEADER_SIZE +
ENUM_NET_ADDR_OFF,              \
                set sr4 [sr3,l],
                                \
                set sr3 sr3 + 4,
                                \
                ADDR_LOOP1, ADDR_LOOP1, ADDR_LOOP1
                                        \
        )

#define handle_stat_reply
                                \
        (
                                                        \
                enum_net_addr, handle_enum_net_addr or 1
                \
        )

/* or (enum_conn_info, handle_enum_conn_info) or \
```

```
	(get_server_info, handle_get_server_info) or (get_routers_info,
handle_get_routers_info) or \
	(get_known_info, handle_get_known_info) or (get_sources_info,
handle_get_sources_info) or \
	(lan_config_info, handle_lan_config_info)) */

#define get_inet_addr_rep
                                \
	(
                                                        \
		sr1 = FUNC_GET_INET_ADDR,
                                        \
		set sr3 TCPDATA+NCP_REPLY_HEADER_SIZE,
NETWARE_PREDICT(sr3)                    \
	)

#define netware_rep_match
                                \
	(
                                                        \
		tcp, sport = NETWARE_NCP_PORT, NCP_REP_TYPE = NCP_REPLY,
		\
		get <rconn, NCP_REP_KEY> from ncp_table to sr1,
ncp_delete_from_table,\
		(nds, handle_nds_reply or 1)
			\
		 or handle_stat_reply or get_inet_addr_rep
		\
	)

#define netware_code
                                \
	(
                                                        \
		netware_req_match
                                        \
	) or (
                                                \
		netware_rep_match
                                        \
	);
```

Stateful D N S

```
#define dns_verification_code
                                        \
        (
                                                        \
                udp or tcp,
                                                \
                sport = 53 or dport = 53,
                                        \
                set sr1 call KFUNC_DNS_FILTER<>, sr1 = 0,
                        \
                LOG(long, LOG_NOALERT, 0), drop
                                        \
        );

/*************************************************************************
*****
 *                                                        *
 *                        CVP                                    *
 *                                                        *

*************************************************************************
****/

#define CVP_MAGIC           0x43565020              /*"CVP "*/
#define cvp_type        [(TCPDATA+6):2,b]
#define cvp_request  ((cvp_type) = 0x1)
#define cvp_response ((cvp_type) = 0x2)
#define cvp_result   ((cvp_type) = 0x3)
#define cvp_drop        ((cvp_type) = 0x4)
#define IP_rec          [(TCPDATA+16):4,b]
#define IP_send                 [(TCPDATA+20):4,b]
#define port_rec        [(TCPDATA+24):2,b]
#define port_send       [(TCPDATA+26):2,b]

#define cvp_first
                                                \
        (dport=FW1_cvp, tcp, packetlen > 47,cvp_request,
        \
         record <src,CVP_MAGIC,dst,dport,ip_p;0,0> in pending)


#define cvp_intercept
                                \
        (sport=FW1_cvp,
                                                \
          ((packetlen > 67,                                                     \
```

```
                ((cvp_response,
                                                \
                        <dst,CVP_MAGIC,src,sport,ip_p> in pending,
                        \
                    record <dst,CVP_MAGIC,IP_rec,port_rec,ip_p;0,0> in pending,
        \
                         record <dst,CVP_MAGIC,IP_send,port_send,ip_p;0,0> in
pending)))))


#define cvp_control_connection cvp_first; cvp_intercept


#define cvp_data_connection
                            \
        (tcp, <src,CVP_MAGIC,dst,dport,ip_p> in pending,
        \
        ((not_first, delete <src,CVP_MAGIC,dst,dport,ip_p> from pending)
        \
        or
                                                \
        (set sr3 0, RECORD_CONN(0xffffff01), IMPLIED_LOG)))


#define cvp_data_and_control                            \
    ((dport=FW1_cvp, tcp, set sr3 0, RECORD_CONN(0xffffff02), IMPLIED_LOG)
        \
         /*control connection*/
                                \
          or
                                                \
    (cvp_data_connection))



/*********************************************************************
Properties Generated Macros
*********************************************************************/

#include "snmp.def"

define accept_domain_udp { accept dport = SERV_domain, udp, set sr3 0,
RECORD_CONN(0xffffff03), IMPLIED_LOG };
define accept_domain_tcp { accept dport = SERV_domain, tcp, set sr3 0,
RECORD_CONN(0xffffff04), IMPLIED_LOG };
```

```
define accept_rip { accept sport = SERV_rip, dport = SERV_rip, udp,
0,RECORD_CONN(0xffffff05), IMPLIED_LOG };


define stateful_icmp {
#ifdef STATEFUL_ICMP
                (
        (icmp_type not in icmp_requests,
                        icmp_type not in icmp_replies,
                        icmp_type not in icmp_errors)
                or
        (icmp_type in icmp_requests,
                                record <src,icmp_id,dst> in icmp_connections)
                                                        or
        (icmp_type in icmp_replies, <dst,icmp_id,src> in icmp_connections)
                or
                        (icmp_type in icmp_errors, dst = icmp_ip_src,
                                (icmp_ip_p != PROTO_tcp, icmp_ip_p != PROTO_udp,
                                 icmp_ip_p != PROTO_icmp)
                                or
                                (icmp_ip_p = PROTO_tcp or icmp_ip_p = PROTO_udp,
                                        (<icmptcpudprconn> in connections
                                  or
                                        <icmptcpudpconn> in connections)
                                )
                                or
                                (icmp_ip_p = PROTO_icmp,
                                        <icmp_ip_src,icmp_icmp_id,icmp_ip_dst> in
icmp_connections
                                )
#ifdef TCP_FASTMODE_ACTIVE
                                or
                                (icmp_ip_p = PROTO_tcp, icmp_th_dport in
tcp_fastmode_services
                                )
#endif
                        )
                )
#else
   1
#endif
};


define inspect_icmp {
   icmp,
```

```
            (
                        (stateful_icmp)
                         or
                        (
#ifdef STATEFUL_ICMP_LOG
                                    set sr1 0, log icmp_badip_form,
#endif
                                    drop
                        )
                  )
};


define accept_icmp {
   accept icmp,
            icmp_type != ICMP_REDIRECT,
#ifndef NO_ENCRYPTION_FEATURES
                  (ACCEPT_CLIENT_ENCRYPTION(0) or 1),
#if ACCEPT_DECRYPT_ENABLE
                  ((direction = 0, ACCEPT_DECRYPT(0)) or 1),
#else
                  IS_NOT_DECRYPTED(0),
#endif
#endif
                  IMPLIED_LOG
};


#ifdef GATEWAY_RULES_INBOUND
#define accept_outgoing {
                                    \
         outbound all@all accept
                                          \
                  set sr3 0, RECORD_CONN(0xffffff06),
                                    \
                  (<[12]> in FWHOST_IP_ADDRS, IMPLIED_LOG) or 1;
                        \
}
#else
#define accept_outgoing {
                                    \
         outbound all@all accept
                                          \
                  <[12]> in FWHOST_IP_ADDRS,
                                          \
```

```
                set sr3 0, RECORD_CONN(0xffffff06), IMPLIED_LOG;
                \
}
#endif

#define block_reverse_tcp
#define block_reverse_udp


#define accept_fw1_connections_first cvp_control_connection

#define accept_fw1_ufp
                                        \
        either bound all@all accept
                                \
                src in firewalled_list, dst in ufp_servers_list,
        \
                dport = FW1_ufp, tcp, set sr3 0, RECORD_CONN(0xffffff07),
IMPLIED_LOG

#define accept_fg1_ela
                                        \
        either bound all@all accept
                                \
                dport = FW1_ela, tcp,
                                \
                src in floodgated_list, dst in management_list,
                \
                set sr3 0, RECORD_CONN(0xffffff07), IMPLIED_LOG

#define accept_fw1_connections1 {
                        \
        either bound all@all accept
                                \
                dst in firewalled_list,
                                \
                (
                                                \
                        (dport = FWD_SVC_PORT or dport = FWD_LOG_PORT, tcp,
                        \
                         src in firewalled_list)
                        \
                                or
                                                \
                        (dport = FWD_TOPO_PORT or dport = FWD_KEY_PORT, tcp)
                \
```

```
                ), set sr3 0, RECORD_CONN(0xffffff08), IMPLIED_LOG;
#define  accept_fw1_connections2
                        \
        either bound all@all accept
                                \
                dport = FWM_SVC_PORT, tcp,
                                        \
                src in gui_clients_list, dst in management_list,
        \
                set sr3 0, RECORD_CONN(0xffffff08), IMPLIED_LOG;
                \
        either bound all@all accept
                                \
                src in firewalled_list,  dst in cvp_servers_list,
        \
                cvp_data_and_control;
                                        \
        accept_fw1_ufp;
                                                \
        accept_fg1_ela;
#define accept_fw1_connections3
                        \
        either bound all@all accept
                                \
        (
                                                        \
                (dport = ISAKMPD_DPORT, udp or tcp,
                                \
                 src in firewalled_list or dst in firewalled_list) or
        \
                (sport = ISAKMPD_SPORT, udp or tcp, src in firewalled_list)
        \
        ) , set sr3 0, RECORD_CONN(0xffffff08);
                        \
        accept_fw1_rdp;
                                                \
}
#define accept_fw1_connections accept_fw1_connections1
        \
                accept_fw1_connections2 accept_fw1_connections3

#define enable_radius_queries {
                                \
        either bound all@all accept
                                \
```

```
                udp,
                                                \
                src in firewalled_list,
                                \
                <dst,dport> in radius_servers_list,
                        \
                set sr3 0, RECORD_CONN(0xffffff09), IMPLIED_LOG;
                \
}

#define enable_tacacs_queries {
                                \
        either bound all@all accept
                                \
                src in firewalled_list,
                                \
                <dst,dport,ip_p> in tacacs_servers_list,
                \
                set sr3 0, RECORD_CONN(0xffffff0c), IMPLIED_LOG;
                \
}

#define enable_ldap_queries {
                                \
        either bound all@all accept
                                \
                tcp,
                                                \
                src in firewalled_list,
                                \
                <dst,dport> in ldap_servers_list,
                        \
                set sr3 0, RECORD_CONN(0xffffff0b), IMPLIED_LOG;
                \
}


#define enable_load_agent_queries {
                        \
    outbound all@all accept
                \
                dport = load_agent_port, udp,
                        \
                src in firewalled_list,
                                \
```

```
                dst in servers_list,
                        \
                set sr3 0, RECORD_CONN(0xffffff0a), IMPLIED_LOG;
                \
}

#ifndef IPOPTNS_LOG
#define IPOPTNS_LOG 1
#endif

#defineHOST_COUNT_LOG(src)
                                \
        log <"![alert]"> host_count_format

#define COUNT_HOST(external_table, count_table)
                \
        (
                                                \
                (interface in external_table)
                        \
                        or
                                                \
                (record <src> in count_table)
                        \
                        or
                                                \
                (
                                                \
                        (src in forbidden_tab)
                                \
                                or
                                                \
                        ((record <src> in forbidden_tab), HOST_COUNT_LOG(src))
                \
                                or
                                                \
                        1,
                                                \
                        drop
                                        \
                )
                                                \
        )

#define COUNT_HOST_CES(count_table)
                                \
```

```
(
                                                \
        (ifid in external_ces_if_list)
                \
                or
                                                \
        (record <src> in count_table)
                \
                or
                                                \
        (
                                                \
                (src in forbidden_tab)
                        \
                        or
                                                \
                ((record <src> in forbidden_tab), HOST_COUNT_LOG(src))
        \
                        or
                                                \
                1,
                                                \
                drop
                                        \
        )
                                                \
)
```

PROXY CONNECTIONS MECHANISM

```
deffunc PROXY_DO(act) {
        set sr3 0,
        (
                (
                        act & 2,
        /* ACCOUNT */
                        set sr3 IS_TRACKED_UNK,
                        (
                                (
                /* ACCOUNT + INSIDE */
                                        act & 4,
                                        direction = 0,
                                        <conn> not in tracked,
                                        record <conn;date+tod,1,packetlen,-
(direction+1),0,-1,
```

```
        date+tod @TCP_TIMEOUT> in tracked
                                ) or
                                (
                                        direction = 1,
                                        <conn> not in tracked,
                                        record <conn;date+tod,1,packetlen,-
(direction+1),0,-1,

        date+tod @TCP_TIMEOUT> in tracked
                                )
                        )
                )
                or 1
        ),
        (
                (
                   !(act & 1),
        /* ACCEPT */
                        RECORD_CONN(-(direction+1)),
                        accept
                )
#ifndef NO_ENCRYPTION_FEATURES
                or
                (
                        act & 1,
        /* ENCRYPT */
                        ENCRYPTION(-(direction+1)),
                        drop
                )
#endif
        )
 };

/*
 *  This code does the action neccesary for a connection which is suppossed
 *  to be proxied , I.e. broken into two connections (using AT folding).
 *  The outbound direction code is different because in that case we put the
 *  table in the entry from user mode , not knowin yet from which interface
 *  we would get out and hence the srcaddr is not known , however in the kernel
 *  it is easily checked the the src addr is equal to the interfaces addr.
 */
#define accept_proxied_conns
                                \
(
                                                        \
```

```
        (
                                                        \
                direction=1,
                                        \
                src=ifaddr,
                                                \
                get <sconn> from proxied_conns to sr10,
                        \
                set r_entry MATCH_BY_PROTOCOL,
                                        \
                set r_connarg 0,
                                        \
                PROXY_DO(PVAL_ACT(sr10))
                                        \
        ) or
                                                \
        (
                                                        \
                direction=0,
                                        \
                get <conn> from proxied_conns to sr10,
                        \
                set r_entry MATCH_BY_PROTOCOL,
                                        \
                set r_connarg 0,
                                        \
                PROXY_DO(PVAL_ACT(sr10))
                                        \
        )
                                                        \
)

#ifdef NON_SYN_RULEBASE_MATCH_LOG
#if LOG_TIMEOUT > 0
#define NON_SYN_RULEBASE_MATCH_LOG_CMD
                                        \
                (
                                                        \
                        (
                                                        \
                                <ip_p,src,dst,sport,dport,0> in logged
                        \
                        ) or (
                                                \
                                record <ip_p,src,dst,sport,dport,0> in logged,
                \
```

```
                                set sr10 RCODE_TCP_EST, set sr11 0, set sr12 0, set sr1
0,      \
                                log bad_conn
                                        \
                        ) or 1
                                                \
                )
#else
#define NON_SYN_RULEBASE_MATCH_LOG_CMD
                                        \
                (
                                                        \
                        (
                                                        \
                                set sr10 RCODE_TCP_EST, set sr11 0, set sr12 0, set sr1
0,      \
                                log bad_conn
                                        \
                        ) or 1
                                                \
                )
#endif
#else
#define NON_SYN_RULEBASE_MATCH_LOG_CMD (1)
#endif

#ifndef ALLOW_NON_SYN_RULEBASE_MATCH
#define start_rule_base_code
                        \
        (
                                                        \
                tcp,
                                                \
                first
                                                \
                        or
                                                        \
                <conn> in old_connections
                                \
                        or
                                                        \
                (src in firewalled_list, dst in firewalled_list)
        \
                        or
                                                        \
```

```
                (direction = 1, <[12]> in FWHOST_IP_ADDRS)
                        \
                        or
                                                        \
                (NON_SYN_RULEBASE_MATCH_LOG_CMD, vanish)
                                \
        ) or 1,
                                                \
        set r_user 0;
#else
#define start_rule_base_code
                        \
        set r_user 0;
#endif

#endif /* __base_def__ */
```

Authentication

Here are describtions of all the 3 types of checkpoint authentication modes.

```
#ifndef NO_AUTH_FEATURES

#include "fwconn.h"

#define AUTH_TRAP                    0x80000000
#define AUTH_LOG_TRAP  0x40000000

#define URL_LOG_CONF    URLOG_LOG_FIRST_SECURE
```

User (Transparent) Authentication

```
#define MOD_counter
                                        \
(
                                                \
        (
                                                \
                sr5 = sr6, set sr6 1
                        \
        )
                                                \
        or
                                                \
        (
                                                \
                set sr6 sr6+1
                                \
        )
                                                \
)
                                                \


#define SET_PORT
                                \
(
                                                \
        (
                                                \
```

```
                TABLE_NOT_EMPTY(client_was_auth),
                        \
                <src,dport> in client_was_auth,
                        \
                get <src,dport> from client_was_auth to sr7,
        \
                set sr4 sr7
                                        \
        )
                                                \
        or
                                                \
        (
                                                \
                get <dport,ip_p,sr6> from auth_services to sr7,
        \
                MOD_counter,
                                        \
                record <dport,ip_p;0,sr5,sr6> in auth_services,
        \
                record <src,dport;sr7> in client_was_auth,
        \
                set sr4 sr7
                                        \
        )
                                                \
)

#define PASS_USERAUTH(rule)
                                        \
(
                                                \
        (
                                                \
                direction=1,
                                        \
                reject
                                        \
        )
                                                \
        or
                                                \
        (
                                                \
```

```
                (get <dport,ip_p> from auth_services to sr4 or reject),
        \
                (
                                                \
                        tcp, first,
                                        \
                        (
                                                \
                                        ((sr4 = 0,SET_PORT) or 1),
                        \
                                        call KFUNC_XLATE_FOLD<ifaddr,sr4,0>
                        \
                                or
                                                \
                                        reject
                                        \
                        )
                                                \
                )
                                                \
        )
                                                \
)
```

Client Authentication

```
#define CLAUTH_LAST_CONN_MAGIC 0x12121212

#define ENABLE_CLNTAUTH(rule)
                                        \
                (dport = auth_services[dport,ip_p])

#define CLNTAUTH_TAB(rule,pnum,isrpc) (
                                \
                (direction = 1,
                                        \
                        (<rule,src,dst,pnum,ip_p,isrpc> in client_auth)
                \
                        or
                                                \

        (<rule,src,dst,pnum,ip_p,isrpc,CLAUTH_LAST_CONN_MAGIC> in
client_auth, \
                        delete
<rule,src,dst,pnum,ip_p,isrpc,CLAUTH_LAST_CONN_MAGIC> \
```

```
                        from client_auth)
                                \
        )
                                        \
        or
                                        \
        (direction = 0,
                                \
                <rule,src,dst,pnum,ip_p,isrpc> in client_auth,
        \
                get <rule,src,dst,pnum,ip_p,isrpc> from client_auth to sr10,\
                set r_user call KFUNC_KBUF_DUP <sr12>,
                        \
                (
                                        \
                        (sr10 > 1, (
                                \
                                (sr11 > 0, record <rule,src,dst,pnum,ip_p,isrpc;
    \
                                                sr10 - 1, sr11,sr12@sr11> in
client_auth)    \
                                or
                                        \
                                modify <rule,src,dst,pnum,ip_p,isrpc;sr10 - 1,
0,sr12>         \
                                        in client_auth)
                                \
                        )
                                        \
                        or (
                                \
                                delete <rule,src,dst,pnum,ip_p,isrpc> from
client_auth,    \
                                record
<rule,src,dst,pnum,ip_p,isrpc,CLAUTH_LAST_CONN_MAGIC@15> \
                                        in client_auth
                                \
                        )
                                        \
                        or 1
                                \
                ),
                                        \
                record <conn ; r_user> in client_auth_username,
    \
```

```
                        TRAP_FLG(client_auth_log, rule, AUTH_LOG_TRAP)
                                \
                )
                                                        \
)


deffunc CLNTAUTH_CALL_AUTHAGENT(rule) {
                        (
                                <conn> in session_requests
                                                        or
                          (
                                        record <conn> in session_requests,
                                        TRAP_FLG(auth_invoke, rule, AUTH_TRAP),
hold
                                ), drop
                        )
};

deffunc CLNTAUTH_CALL_SSOAGENT(rule) {
                        (
                                set sr9 0,
                                (
                                        get <conn> from sso_requests to sr9
                                ) or (
                                        record <conn;0> in sso_requests,
                                        TRAP_FLG(clauth_invoke, rule, AUTH_TRAP),
hold, drop
                                ),
                                sr9 = 0, drop
                        )
};

deffunc CLNTAUTH_MUST_FOLD(rule) {
                        (<rule,src> in autoclntauth_fold,
                        delete <rule, src> from autoclntauth_fold,
                        PASS_USERAUTH(rule))
};

deffunc PASS_CLNTAUTH_SPECIFIC(rule) {
                (

                (CLNTAUTH_TAB(rule,dport,0)

                        or
```

```
                (<rule,src,dst,pm_prog,ip_p,1> in client_auth,rpc_getport(pm_prog))
                        or

                (CLNTAUTH_TAB(rule,cb_prog,1), rpc_insession(cb_prog))

                        or

                (CLNTAUTH_TAB(rule,pm_prog,1), rpc_callit(pm_prog))

                ))
};


#define pass_clntauth_record (
                                \
        (sr2 = 0, record <rule,src;sr10 - 1,sr11,sr12@sr11> in client_auth)            \
 )

#define pass_clntauth_modify (
                                \
        (sr2 = 0, modify <rule,src;sr10 - 1,sr11,sr12> in client_auth)
        \
 )

#define pass_clntauth_delete (
                                \
        (sr2 = 0, delete <rule,src> from client_auth),
        \
        record <rule, src, CLAUTH_LAST_CONN_MAGIC@15> in client_auth
        \
 )

deffunc PASS_CLNTAUTH(rule)     {
        (
                PASS_CLNTAUTH_SPECIFIC(rule)
                                or
                (direction = 1, (<rule,src> in client_auth) or

        (<rule,src,CLAUTH_LAST_CONN_MAGIC> in client_auth,
                                                delete
<rule,src,CLAUTH_LAST_CONN_MAGIC> from client_auth)
                )
                                or
                (direction = 0,
                        ((<rule,src> in client_auth, set sr2 0, get <rule,src> from
client_auth to sr10)),
```

```
                        set r_user call KFUNC_KBUF_DUP <sr12>,
                        (
                                (sr10 > 1, (sr11 > 0, pass_clntauth_record) or
pass_clntauth_modify)
                                or
                                (
                                        pass_clntauth_delete,
                                        get <src, 1> from check_alive to sr10,
                                        (
                                                (sr13 > 1, modify <src,1;sr10,sr11,sr12,sr13
- 1> in check_alive)
                                                or
                                                delete <src, 1> from check_alive
                                        )
                                )
                                or 1
                        ),
                        record <conn ; r_user> in client_auth_username,
                        TRAP_FLG(client_auth_log, rule, AUTH_LOG_TRAP)
                )
        )
};

deffunc CHECK_CLNTAUTH_SPECIFIC(rule) {
                (
                        (<rule,src,dst,dport,ip_p,0> in client_auth)
                                or
                        (<rule,src,dst,pm_prog,ip_p,1> in client_auth,
                                        (rpc_getport(pm_prog) or rpc_callit(pm_prog)))
                                or
                        (<rule,src,dst,cb_prog,ip_p,1> in client_auth,
                                        rpc_insession(cb_prog))
                )
};

deffunc CHECK_CLNTAUTH(rule) {
        (<rule,src> in client_auth)
                or
        (direction = 1,<rule,src,CLAUTH_LAST_CONN_MAGIC> in client_auth)
                or
        CHECK_CLNTAUTH_SPECIFIC(rule)
};
```

Session Authentication

After the session authentication daemon completes a successful authentication protocol with the agent, it searches the rulebase permissively for rules matching the authenticated user and the requested connection parameters.
It then fills the following entries in the session_auth table:
for each matching rule 'r_num' the entry <r_num,conn> plus two 'token' entries, <-1,conn>, <-2,conn> for the inbound and outbound directions, respectively.

```
#define PASS_SESSION_AUTH(rule)
                                        \
                (
                                                \
                        (
                                                \
                                <rule,conn> in session_auth,
                        \
                                (
                                                \
                                        (direction = 0, <-1,conn> in session_auth,
                \
                                                delete <-1,conn> from session_auth)
                        \
                                        or
                                                \
                                        (direction = 1, <-2,conn> in session_auth,
                \
                                                delete <-2,conn> from session_auth,
                        \
                                                delete <rule,conn> from session_auth)
                        \
                                ),
                                                \
                                delete <conn, rule | AUTH_TRAP> from trapped
                \
                        ) or (
                                                \
                                rpc_getport(pm_prog)
                                \
                        ) or (
                                                \
                                (
                                                \
                                        <conn> in session_requests
                                \
```

```
                                        or
                                        \
                                <-1,conn> in session_auth
                        \
                                        or
                                        \
                                (
                                        \
                                        record <conn> in session_requests,
                \
                                        TRAP_FLG(auth_invoke, rule,
AUTH_TRAP), hold        \
                                )
                                        \
                        ), drop
                                        \
                )
                                        \
        )

/* URL logging without UFP */
#define DO_URL_LOG_SVC(rule, port, flags)
                \
    (tcp, dport = port,
                                \
        <conn> in urlog_conns
                                \
        or
                                        \
        record <conn; (th_seq+1), flags | URLOG_NO_UFP, rule, 0> in
urlog_conns \
        or
                                        \
        drop
                                        \
    )

#define URL_LOG_AUTH_SVC(rule, port, log_method)
        \
    (DO_URL_LOG_SVC(rule, port, log_method | URLOG_WITH_AUTH))

#define URL_LOG_SVC(rule, port, log_method)
                \
    (DO_URL_LOG_SVC(rule, port, log_method))
```

```
#define IN_UFP_CACHE(rule)(get <dst,dport,rule> from ipufp_cache to sr10)

#define DO_URL_LOG(rule, LOG_TYPE, category) (
                                \
        <conn> in urlog_conns
                                        \
        or
                                                \
        record <conn;(th_seq+1),URL_LOG_CONF | LOG_TYPE, rule, category> in
urlog_conns \
        or
                                                \
        drop
                                                \
 )

#define DO_AUTH_URL_LOG(rule,
category)(DO_URL_LOG(rule,URLOG_WITH_AUTH,    \
                category))

#define DO_NOAUTH_URL_LOG(rule, category) (DO_URL_LOG(rule,0, category))

#define NOT_IN_UFP_CACHE(rule)(not IN_UFP_CACHE(rule))

#define IS_MATCH_CACHE_UFP(rule)(IN_UFP_CACHE(rule),(sr10 = 1))

#define PASS_AUTH_URL_LOG(rule)                         \
        ((IN_UFP_CACHE(rule),DO_AUTH_URL_LOG(rule, sr11)) or drop)

#define IS_MATCH_LOG_NOAUTH_CACHE_UFP(rule)                     \
        (IS_MATCH_CACHE_UFP(rule),(DO_NOAUTH_URL_LOG(rule, sr11) or
drop))

#define DO_AU_CLAUTH_CACHEUFP(rule,dolog)                       \
        ((NOT_IN_UFP_CACHE(rule) ,PASS_USERAUTH(rule)) or               \
    (IS_MATCH_CACHE_UFP(rule), ((PASS_CLNTAUTH(rule),               \
          ((dolog,PASS_AUTH_URL_LOG(rule)) or 1)) or PASS_USERAUTH(rule))))

#define
AU_CLAUTH_CACHEUFP_LOG(rule)(DO_AU_CLAUTH_CACHEUFP(rule,1))

#define
AU_CLAUTH_CACHEUFP_NOLOG(rule)(DO_AU_CLAUTH_CACHEUFP(rule,0))

#define DO_CLAUTH_CACHEUFP(rule,dolog)                          \
```

```
        (CHECK_CLNTAUTH(rule),(NOT_IN_UFP_CACHE(rule),
PASS_USERAUTH(rule))    \
        or (IS_MATCH_CACHE_UFP(rule), PASS_CLNTAUTH(rule),                    \
        ((dolog,PASS_AUTH_URL_LOG(rule)) or 1)))

#define CLAUTH_CACHEUFP_LOG(rule)(DO_CLAUTH_CACHEUFP(rule,1))

#define CLAUTH_CACHEUFP_NOLOG(rule) (DO_CLAUTH_CACHEUFP(rule,0))

#define DO_SPECIFIC_CLAUTH_CACHEUFP(rule,dolog)                       \
        (CHECK_CLNTAUTH_SPECIFIC(rule),(NOT_IN_UFP_CACHE(rule),
\
         PASS_USERAUTH(rule)) or (IS_MATCH_CACHE_UFP(rule),                   \
         PASS_CLNTAUTH_SPECIFIC(rule),((dolog,PASS_AUTH_URL_LOG(rule))
or 1)))

#define CLAUTH_SPECIFIC_CACHEUFP_LOG(rule)                            \
        (DO_SPECIFIC_CLAUTH_CACHEUFP(rule,1))

#define CLAUTH_SPECIFIC_CACHEUFP_NOLOG(rule)                          \
        (DO_SPECIFIC_CLAUTH_CACHEUFP(rule,0))

/********************
 * X11 Verification *
 ********************/

#define x11verify_code(table)
                                \
                (
                                                        \
                        tcp, dport >= 6000, dport < 6070,
                        \
                        ((
                                                        \
                                <[12]> in FWHOST_IP_ADDRS
                                        \
                        ) or (
                                                \
                                <conn> in table,
                                        \
                                set sr10 table[conn],
                                \
                                (
                                                        \
                                        (sr10 = 0, drop)
                                        \
```

```
                                        or
                                                \
                                (sr10 = 1,
                                        \
                                        (((not_first or (direction = 1)),
                        \
                                                                        delete <conn> from
table) or 1))           \
                                        or
                                                \
                                (sr10 = 2, set sr10 sr1,
                        \
                                 LOG(long,LOG_NOALERT,sr10), drop)
                                \
                                )
                                                \
                        ) or (
                                        \
                                record <conn; 0> in table,
                                \
                                set sr10 100, log trap_execute, hold
                \
                        ))
                                                \
                )

#else    /* NO_AUTH_FEATURES */

#define PASS_USERAUTH(rule)             (1)
#define PASS_CLNTAUTH(rule)             (1)
#define PASS_SESSION_AUTH(rule) (1)

#endif    /* NO_AUTH_FEATURES */
```

Traps.def

```
#include "tcpip.def"
#include "traps.h"

auth_invoke = format AUTH_INVOKE {
	<"magic", "hex", 0x50415254>,
	<"nargs", "int", 11>,
	<"src", ipaddr, src>,
	<"s_port", port, sport>,
	<"dst", ipaddr, dst>,
	<"service", port, dport>,
	<"proto", proto, ip_p>,
	<"isrpc", int, r_entry>,
	<"rpcnum", int, r_connarg>,
	<"rule", int, sr1>,
	<"packetid", int, packetid>,
	<"ifid", int, ifid>,
	<"trap auth_invoke", int, AUTH_INVOKE>
};

auth_intrcpt = format AUTH_INTRCPT {
	<"magic", "hex", 0x50415254>,
	<"nargs", "int", 7>,
	<"src", ipaddr, src>,
	<"s_port", port, sport>,
	<"dst", ipaddr, dst>,
	<"service", port, dport>,
	<"proto", proto, ip_p>,
	<"rule", int, sr1>,
	<"trap auth_intrcpt", int, AUTH_INTRCPT>
};

pmap_fetch = format PMAP_FETCH {
	<"magic", "hex", 0x50415254>,
	<"nargs", "int", 3>,
	<"dst", ipaddr, dst>,
	<"packetid", int, packetid>,
	<"trap pmap_fetch", int, PMAP_FETCH>
};

userc_server_invoke = format USERC_SERVER_INVOKE {
	<"magic", "hex", 0x50415254>,
	<"nargs", "int", 23>,
	<"src", ipaddr, src>,
	<"s_port", port, sport>,
```

```
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"proto", proto, ip_p>,
        <"rule", int, sr1>,
        <"orig_dst", ipaddr, origdst>,
        <"orig_dport", port, origdport>,
        <"packetid", int, packetid>,
        <"entry", int, r_entry>,
        <"connarg", int, r_connarg>,
        <"decrypt", int, sr2>,
        <"ifid", int, ifid>,
        <"isskip", int, wasskipped>,
        <"skippeer", int, skippeer>,
        <"skipme", int, skipme>,
        <"ipsecmethods", int, ipsecmethods>,
        <"ipsecdata", int, ipsecdata>,
        <"xlatesrc", ipaddr, xlatesrc>,
        <"xlatedst", ipaddr, xlatedst>,
        <"xlatesport", port, xlatesport>,
        <"xlatedport", port, xlatedport>,
        <"trap userc_server_invoke", int, USERC_SERVER_INVOKE>
};

encrypt_invoke = format ENCRYPT_INVOKE {
        <"magic", "hex", 0x50415254>,
        <"nargs", "int", 20>,
        <"src", ipaddr, src>,
        <"s_port", port, sport>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"proto", proto, ip_p>,
        <"rule", int, sr1>,
        <"packetid", int, packetid>,
        <"entry", int, r_entry>,
        <"connarg", int, r_connarg>,
        <"interface", int, ifid>,
        <"isskip", int, wasskipped>,
        <"skippeer", int, skippeer>,
        <"skipme", int, skipme>,
        <"ipsecmethods", int, ipsecmethods>,
        <"ipsecdata", int, ipsecdata>,
        <"xlatesrc", ipaddr, xlatesrc>,
        <"xlatedst", ipaddr, xlatedst>,
        <"xlatesport", port, xlatesport>,
        <"xlatedport", port, xlatedport>,
        <"trap encrypt_invoke", int, ENCRYPT_INVOKE>
```

```
};

encrypt_intrcpt = format ENCRYPT_INTRCPT {
        <"magic", "hex", 0x50415254>,
        <"nargs", "int", 7>,
        <"src", ipaddr, src>,
        <"s_port", port, sport>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"proto", proto, ip_p>,
        <"packetid", int, packetid>,
        <"trap encrypt_intrcpt", int, ENCRYPT_INTRCPT>
};

decrypt_invoke = format DECRYPT_INVOKE {
        <"magic", "hex", 0x50415254>,
        <"nargs", "int", 12>,
        <"src", ipaddr, src>,
        <"s_port", port, sport>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"proto", proto, ip_p>,
        <"rule", int, sr1>,
        <"packetid", int, packetid>,
        <"entry", int, r_entry>,
        <"connarg", int, r_connarg>,
        <"interface", int, ifid>,
        <"key", int, r_ckey>,
        <"trap decrypt_invoke", int, DECRYPT_INVOKE>
};

decrypt_invoke_other = format DECRYPT_INVOKE {
        <"magic", "hex", 0x50415254>,
        <"nargs", "int", 12>,
        <"src", ipaddr, src>,
        <"s_port", int, sr1>,
        <"dst", ipaddr, dst>,
        <"service", int, 0>,
        <"proto", proto, ip_p>,
        <"rule", int, sr1>,
        <"packetid", int, packetid>,
        <"entry", int, r_entry>,
        <"connarg", int, r_connarg>,
        <"interface", int, ifid>,
        <"key", int, r_ckey>,
        <"trap decrypt_invoke", int, DECRYPT_INVOKE>
```

```
};

trap_execute = format TRAP_EXECUTE {
        <"magic", "hex", 0x50415254>,
        <"nargs", "int", 8>,
        <"src", ipaddr, src>,
        <"s_port", port, sport>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"proto", proto, ip_p>,
        <"packetid", int, packetid>,
        <"prog num", int, sr10>,
   <"trap execute", int, TRAP_EXECUTE>
};

logical_invoke = format BALANCE_INVOKE {
        <"magic", "hex", 0x50415254>,
        <"nargs", "int", 8>,
        <"src", ipaddr, src>,
        <"s_port", port, sport>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"proto", proto, ip_p>,
        <"rule", int, sr1>,
        <"packetid", int, packetid>,
        <"trap encrypt_invoke", int, BALANCE_INVOKE>
};

client_auth_log = format CLIENT_AUTH_LOG {
        <"magic", "hex", 0x50415254>,
        <"nargs", "int", 15>,
        <"proto", proto, ip_p>,
        <"src", ipaddr, src>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"s_port", port, sport>,
        <"rule", rule, sr1>,
        <"username", int, r_user>,
        <"len", int, packetlen>,
        <"xlatesrc", ipaddr, xlatesrc>,
        <"xlatedst", ipaddr, xlatedst>,
        <"xlatesport", port, xlatesport>,
        <"xlatedport", port, xlatedport>,
        <"interface", int, ifid>,
        <"is_sso", int, sr2>,
        <"trap client_auth_log", int, CLIENT_AUTH_LOG>
```

```
};

clauth_invoke = format CLAUTH_INVOKE {
        <"magic", "hex", 0x50415254>,
        <"nargs", "int", 10>,
        <"src", ipaddr, src>,
        <"s_port", port, sport>,
        <"dst", ipaddr, dst>,
        <"service", port, dport>,
        <"proto", proto, ip_p>,
        <"isrpc", int, r_entry>,
        <"rpcnum", int, r_connarg>,
        <"rule", int, sr1>,
        <"packetid", int, packetid>,
        <"trap clauth_invoke", int, CLAUTH_INVOKE>
};
```

SNMP protocol

```
#define SNMP_VERSION_1     0
#define SNMP_VERSION_2C     1
#define SNMP_VERSION_2     2

#define ASN_CONTEXT                    (0x80)
#define ASN_CONSTRUCTOR          (0x20)

#define GET_REQ_MSG                    (ASN_CONTEXT |
ASN_CONSTRUCTOR | 0x0)
#define GETNEXT_REQ_MSG          (ASN_CONTEXT | ASN_CONSTRUCTOR |
0x1)
#define GET_RSP_MSG                    (ASN_CONTEXT |
ASN_CONSTRUCTOR | 0x2)
#define SET_REQ_MSG                    (ASN_CONTEXT |
ASN_CONSTRUCTOR | 0x3)
#define TRP_REQ_MSG                    (ASN_CONTEXT |
ASN_CONSTRUCTOR | 0x4)

/* SNMPv2 and SNMPv2C only ops */
#define BULK_REQ_MSG      (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x5)
#define INFORM_REQ_MSG          (ASN_CONTEXT | ASN_CONSTRUCTOR |
0x6)
#define TRP2_REQ_MSG      (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x7)

#define ber_sequence          ([UDPDATA:1] = 0x30)
#define snmp_version_offset  \
        (((((([UDPDATA+1:1] & 0x80) >> 7) * 0x7f) & ([UDPDATA+1:1])) + 4)
#define snmp_version          [UDPDATA+snmp_version_offset:1]
#define snmp_community_len[UDPDATA+snmp_version_offset+2:1]
#define snmp_op
        [(UDPDATA+snmp_version_offset+3+snmp_community_len):1]

#define snmp_port                    161

snmp_ro_tab = dynamic refresh sync expires UDP_TIMEOUT;

define snmpv1 {
        udp, dport = snmp_port, ber_sequence, snmp_version = SNMP_VERSION_1
};

define snmpv1_rw { snmpv1 };

define snmpv1_ro {
```

```
        snmpv1, snmp_op != SET_REQ_MSG , record <udpconn> in snmp_ro_tab
};

define snmpv2 {
        udp, dport = snmp_port, ber_sequence, snmp_version = SNMP_VERSION_2
};

#define snmp_ro_code
                                        \
        drop
                                                \
                snmpv1, snmp_op = SET_REQ_MSG,
                                        \
                <udpconn> in connections, <udpconn> in snmp_ro_tab,
                \
                delete <conn> from connections, delete <udpconn> from connections,
        \
                delete <udpconn> from snmp_ro_tab, LOG(long, LOG_NOALERT, 0);
```

H.323 protocol

#ifndef NO_H323

H.323 - General Protocol Support

```
/* Protocol Name: H.323
 * Protocol Description:
 * client starts H.225 TCP connection on port 1720 to server.
 * server sends "port" command to client, for the H.245 TCP connection.
 * On the H.245 connection, each side send the other its port numbers, and the
 * other side acknolodge with its port numbers for the RTP connections.
 */

/* Added to allow the Bay/Nortel platforms to function - they only filter
 * in the inbound direction.
 */
#ifdef FWEMBEDED
#define H323DIRECTION_IN  0
#define H323DIRECTION_OUT 0
#else
#define H323DIRECTION_IN  0
#define H323DIRECTION_OUT 1
#endif

#define ALLOW_T120 1

#define H245_MAGIC                  0x48323435  /* "H245" */
#define HRTP_MAGIC                  0x48525450  /* "HRTP" */
#define H225_PORT           1720    /* static port of H225 */
#define H245_TIMEOUT        600     /* 10 minutes */
#define RTP_TIMEOUT                 600     /* 10 minutes */

/*
 * Possible return values after h.245 tracing.
 */
#define H_NOTHING_FOUND         0x01
#define H_EST_LOG_CHAN_ACK      0x02
#define H_EST_LOG_CHAN          0x04
#define H_EST_LOG_CHAN_ACK_2    0x08
#define H_EST_LOG_CHAN_2        0x10
#define H_EST_LOG_CHAN_T120             0x20
#define H_EST_LOG_CHAN_ACK_T120         0x40
#define H_225_SETUP                             0x80
#define H_225_CONNECT                   0x160
```

```
#define H_BAD_PACKET            0x00

#define H_FOLLOW_225                    0xfffffffc
#define H_FOLLOW_245                    0xfffffffb
#define H_INIT_225                      0xfffffffd
#define H_INIT_245                      0xffffffff

/*
 * kernel function parameter value, indicating
 * what port the inspect wants to retreive.
 */
#define H_REQ_RTCP      0
#define H_ACK_RTCP      1
#define H_ACK_RTP       2
#define H_ACK_RTCP_2    3
#define H_ACK_RTP_2     4
#define H_REQ_RTCP_2    5
#define H_REQ_T120              6
#define H_ACK_T120              7
#define H_SETUP_SRC             8
#define H_SETUP_DST             9
#define H_CONNECT               10


deffunc GET_H323_VAL(WhatDetected, WhatVal) {
        (sr1 & WhatDetected) =      WhatDetected,
         set sr3 0, set sr4 0,
         set sr2 call KFUNC_FOLLOW_H323<WhatVal,3,4>,
         sr2, set sr2 ((sr3.[0:1]) << 8) + sr3.[1:1],
         (sr2 = 1503 or NOTSERVER_TCP_PORT(sr2) or reject),
         (sr4,
          set sr13 ((sr4.[0:1])<<8)+((sr4.[1:1])),
          set sr4 ((sr4.[2:1]) << 8) + sr4.[3:1],
          set sr13 sr4+(sr13<<16)) or 1
};

deffunc SAVE_H245_PORT_NEW(port,ip,dir,proto,key,flags) {
        (
                set sr4 call KFUNC_H323_SAME_GW<src,ip>,
                (sr4 or (WRONG_HOST_LOG,reject)),
                (ENTRY_TYPE(sr11) = CONN_TCP,
                        ((proto = PROTO_tcp, set sr14 CONN_TCP)
                         or
                        (proto = PROTO_udp, set sr14 CONN_UDP)))
                or
                 (set sr14 ENTRY_TYPE(sr11)),
```

```
                (dir = 1,
                        record <ip,port,dst,HRTP_MAGIC,proto;key,sr14,flags

        @RTP_TIMEOUT> in pending,
                        (ENTRY_TRACKED(flags),
                         record <0,ip,port,dst,HRTP_MAGIC,proto;sr5,sr6,sr7,sr8,sr9

        @RTP_TIMEOUT> in tracked)
                        or 1
                )
                or
                (
                        record <dst,HRTP_MAGIC,ip,port,proto;key,sr14,flags

        @RTP_TIMEOUT> in pending,
                        (ENTRY_TRACKED(flags),
                         record <0,dst,HRTP_MAGIC,ip,port,proto;sr5,sr6,sr7,sr8,sr9

        @RTP_TIMEOUT> in tracked)
                        or 1
                )
        )
};


#ifndef NO_XLATION

/* This is for the Connect message - the port command. */
#define TRANSLATE_H225_NEW(port,offset)
                        \
        (call KFUNC_XLATE_ANTICIPATE <dst,0,0,port,PROTO_tcp,
        \
                                BUILD_CONT_REV(0,1), RTP_TIMEOUT,0,offset,offset
- 4>)

#define TRANSLATE_H245(port,offset,proto)
                        \
         ((r_cdir=2,call KFUNC_XLATE_ANTICIPATE <dst,0,0,port,proto,
        \
                                BUILD_CONT_REV(0,1), RTP_TIMEOUT,0,offset,offset
- 4 >) or        \
         (call KFUNC_XLATE_ANTICIPATE <src,port,dst,0,proto,
                \
                                BUILD_CONT_REV(0,0), RTP_TIMEOUT,0,offset,offset
- 4>))
```

```
/* This is for the Setup message - we need to change the identifiers
 * of the connection inside the data of the packet.
 */
#define TRANSLATE_H225_SETUP(cr0,cr1,off)
        \
        (call KFUNC_XLATE_ANTICIPATE <conn,
                        \
                                BUILD_CONT_REV(cr0,cr1), RTP_TIMEOUT,0,off,off -
4>)
#else
#define TRANSLATE_H225_NEW(port,offset)                 1
#define TRANSLATE_H245(port,offset,proto)               1
#define TRANSLATE_H225_SETUP(cr0,cr1,off)               1
#endif




/*
 * intercept H.225 CONNECT/SETUP message, and record the
 * H.245 dynamic connection that we should accept.
 * If the H.225 connection started from A to B, We are only
 * interested in tracking packets going from B to A, since the
 * CONNECT/SETUP message is in that direction.
 * For first packet of H.225 (B->A) we init the align machine.
 * For every packet of H.225 (B->A) we call the align machine.
 * KFUNC_FOLLOW_H225 return values:
 * 0 = Bad Packet
 * 1 = H.225 connect not found in packet
 * not 0, not 1 - the offset of the port in CONNECT/SETUP message
 */


#define FIND_SETUP_NEW
                                        \
        (
                                                \
                tcp, first, dport = H225_PORT,
                                \
                direction = H323DIRECTION_OUT,
                                        \
                call KFUNC_FOLLOW_H323<H_INIT_225,direction>
                        \
        )
```

```
#define INIT_H225
                                        \
        (
                                                        \
                syn, ack,
                                                \
                call KFUNC_FOLLOW_H323<H_INIT_225,direction>
                        \
        )


#define RECORD_H245_1_NEW
                                \
        (
                                                        \
                (r_cdir = 2, sport = H225_PORT) or (r_cdir =1,dport = H225_PORT),
        \
                tcp,
                                                \
                (
                                                        \
                         INIT_H225 or 1
                                                \
                ),
                                                        \
                set sr1 call KFUNC_FOLLOW_H323<H_FOLLOW_225,direction>,
                        \
                (sr1 != H_BAD_PACKET) or reject,
            \
                (
                                                        \
                        sr1 != H_NOTHING_FOUND,
                                \
                        (GET_H323_VAL(H_225_CONNECT,H_CONNECT),
                                \
                        record <dst,H245_MAGIC,src,sr2,ip_p;DUP_KEY(r_ckey),
                \
                                                r_ctype, r_cflags@H245_TIMEOUT> in
pending,                        \
                        TRANSLATE_H225_NEW(sr2,sr3)) or
#define RECORD_H245_2_NEW
                                \
                        (GET_H323_VAL(H_225_SETUP, H_SETUP_SRC),
                                \
                        TRANSLATE_H225_SETUP(1,0,sr3),
                                \
```

```
                        GET_H323_VAL(H_225_SETUP, H_SETUP_DST),
                                        \
                        TRANSLATE_H225_SETUP(2,0,sr3)),
                                        \
                        (ENTRY_TRACKED(r_cflags),
                                        \
                        record
<0,dst,H245_MAGIC,src,sr2,ip_p;rconn@H245_TIMEOUT>                  \

                                in tracked)      \
                        or 1
                                                \
                ) or 1,
                                                \
                accept_fwz_as_clear(r_ctype)
                        \
        )

#define RECORD_H245_NEW FIND_SETUP_NEW; RECORD_H245_1_NEW
RECORD_H245_2_NEW


/*
 * on first H.245 packet move the connection to connections table
 * with key and type of H.225 (they are in the same direction).
 * Keep the H.245 connection in pending table - to intercept the
 * ESTABLISH_LOGICAL_CHANNEL_ACK commands.
 * Initialize the align machine that will look for ESTABLISH_LOGICAL_CHANNEL
 * req/ack
 * This packet is accepted by  IS_ACCEPTED_A  if clear, or connection table
 *  if encrypted
 */

#define ACCEPT_H245_FWD1_NEW
                                        \
                tcp, first,
                                                \
                TABLE_NOT_EMPTY(pending),
                                        \
                get <src,H245_MAGIC,dst,dport,ip_p> from pending to sr10,
        \
                NOT_TCP_FASTMODE_PORT(sport,0) or reject,
                        \
                record
<conn;DUP_KEY(sr10),ENTRY_TYPE(sr11),SPOOF_CACHE_EMPTY |          \
```

```
IS_ACCEPTED_A|TRACKED_TRANS(sr12)|MORE_INSPECTION|RECORD_SRC(0
x88) \

        @TCP_TIMEOUT> in connections, \
                (ENTRY_TRACKED(sr12),
#define ACCEPT_H245_FWD2_NEW
                                        \
                ACCOUNT_MATCH(H245_MAGIC,0,TCP_TIMEOUT)) or 1,
                                \
                ((direction = H323DIRECTION_OUT or <[16]> in
FWHOST_IP_ADDRS),               \
                call KFUNC_FOLLOW_H323<H_INIT_245,direction>) or 1,
                        \
           update_tcp_flags(sr11,1)

#define ACCEPT_H245_FWD_NEW ACCEPT_H245_FWD1_NEW
ACCEPT_H245_FWD2_NEW

#define ACCEPT_H245_BAK_NEW
                                                \
                tcp, syn, ack,
                                        \
                TABLE_NOT_EMPTY(pending),
                                        \
                get <dst,H245_MAGIC,src,sport,ip_p> from pending to sr10,       \
                ((direction = H323DIRECTION_IN or <[12]> in FWHOST_IP_ADDRS),
        \
                call KFUNC_FOLLOW_H323<H_INIT_245,direction>) or 1


#define ACCEPT_H245_NEW ACCEPT_H245_FWD_NEW;
ACCEPT_H245_BAK_NEW

/* RECORD_RTP macro broken to A few macros
 * intercept the ESTABLISH_LOGICAL_CHANNEL_ACK command (on the H.245
 * connection).
 * Record both the RTP (audio/video) and RTCP (control) (+1) connections.
 * The align machine looks for ESTABLISH_LOGICAL_CHANNEL_ACK in both
direction.
 * KFUNC_FOLLOW_H245 return codes:
 * 0 Bad packet
 * 1 Nothing found in packet
 * EST_LOG_CHAN_ACK (2) - Establish logical channel ack found in packet
 * EST_LOG_CHAN (4) - Establish logical channel found in packet
 * the last 2 may arrive together!
```

```
 * KFUNC_GET_RTP first argument:
 * 0 - get offset of RTCP port in open logical channel msg
 * 1 - get offset of RTCP port in open logical chan ack msg
 * 2 - get offset of RTP port in open logical chan ack msg
 */

#define FOLLOW_H245_NEW
                                                        \
        (
                                                                \
                set sr1 (call KFUNC_FOLLOW_H323<H_FOLLOW_245,direction>),
                \
                (sr1 != H_BAD_PACKET) or reject,
                        \
                sr1 != H_NOTHING_FOUND
                                                        \
        )


deffunc H245_PROCESS_MSG(WhatDetected, WhatVal, proto) {
        GET_H323_VAL(WhatDetected, WhatVal),
        (
                r_cdir=2, TRANSLATE_H245(sr2,sr3,proto),
                GET_H323_VAL(WhatDetected, WhatVal)
        ) or 1,
        SAVE_H245_PORT_NEW(sr2,sr13,r_cdir,proto,DUP_KEY(sr10),sr12),
        (r_cdir=1, TRANSLATE_H245(sr2,sr3,proto)) or 1
};


#ifdef ALLOW_T120
#define GET_H245_T120_ACK
                                                \
(
                                                                \
                GET_H323_VAL(H_EST_LOG_CHAN_ACK_T120, H_ACK_T120),
                                \
                ((direction = H323DIRECTION_OUT,
                                \
                                <src,sr2,dst,HRTP_MAGIC,PROTO_tcp> in pending)
                        \
                  or
                                                        \
                (direction = H323DIRECTION_IN,
                                \
```

```
                        <dst,HRTP_MAGIC,src,sr2,PROTO_tcp> in pending)),
            \
            H245_PROCESS_MSG(H_EST_LOG_CHAN_ACK_T120,
H_ACK_T120, PROTO_tcp) \
 )

#define GET_H245_T120_REQ
                                    \
(
                                                \
            H245_PROCESS_MSG(H_EST_LOG_CHAN_T120, H_REQ_T120,
PROTO_tcp)      \
            or 1
                                        \
 )
#else
#define GET_H245_T120_ACK 1
#define GET_H245_T120_REQ 1
#endif

#define RECORD_RTP1_NEW
                        \
        (
                                                        \
            r_cdir, tcp, not_first, TABLE_NOT_EMPTY(pending),
                        \
            (
                                                        \
                r_cdir=1,
                                                \
                get <src,H245_MAGIC,dst,dport,ip_p> from pending to sr10
                \
            ) or (
                                                \
                r_cdir=2,
                                                \
                get <dst,H245_MAGIC,src,sport,ip_p> from pending to sr10
                \
            ),
#define RECORD_RTP2_NEW
                                                \
            (
                                                        \
                FOLLOW_H245_NEW,
                        \
```

```
                    (ENTRY_TRACKED(sr12),
                                                    \
                    (r_cdir = 1, get <0,conn> from tracked to sr5)
                            \
                     or
                                                    \
                    (get <0,rconn> from tracked to sr3)) or 1,
                            \
                (
                                                    \
                            (
        \
                                    GET_H245_T120_REQ,
                                                    \

        H245_PROCESS_MSG(H_EST_LOG_CHAN_ACK, H_ACK_RTP,
PROTO_udp),         \

        H245_PROCESS_MSG(H_EST_LOG_CHAN_ACK, H_ACK_RTCP,
PROTO_udp),\

        H245_PROCESS_MSG(H_EST_LOG_CHAN_ACK_2, H_ACK_RTP_2,
PROTO_udp),\

        H245_PROCESS_MSG(H_EST_LOG_CHAN_ACK_2, H_ACK_RTCP_2,
PROTO_udp),\
                                    0
                                                            \
                            ) or (
                                                    \
                                    H245_PROCESS_MSG(H_EST_LOG_CHAN,
H_REQ_RTCP, PROTO_udp),             \
                                    H245_PROCESS_MSG(H_EST_LOG_CHAN_2,
H_REQ_RTCP_2, PROTO_udp),   \
                                    0
                                                            \
                            ) or (
                                                    \
                                    GET_H245_T120_ACK
                                                    \
                            )
                                                            \
                    )
                                                            \
            ) or 1,
                                                    \
```

```
                accept_fwz_as_clear(r_ctype)
                                        \
        )

#define RECORD_RTP_NEW RECORD_RTP1_NEW RECORD_RTP2_NEW

/*
 * accept RTP and RTCP connections
 *
 */

#define ACCEPT_RTP1
                                        \
                udp,
                                        \
                TABLE_NOT_EMPTY(pending),
                                        \
                (get <src,HRTP_MAGIC,dst,dport,ip_p> from pending to sr10,
        \
                record <conn;0,sr11 | _UDP_ESTABLISHED,
                        \

        SPOOF_CACHE_EMPTY|TRACKED_TRANS(sr12)|RECORD_SRC(0xa8)
        \
                                                @RTP_TIMEOUT> in
connections,                            \
                record <udpconn;DUP_KEY(sr10),sr11 | _UDP_ESTABLISHED,
                \

        SPOOF_CACHE_EMPTY|TRACKED_TRANS(sr12)|RECORD_SRC(0xa8)
        \
                                                @RTP_TIMEOUT> in
connections,                            \
                delete <src,HRTP_MAGIC,dst,dport,ip_p> from pending,
        \
                set sr9 1,
                                        \
                (ENTRY_TRACKED(sr12),
                                        \
                ACCOUNT_MATCH(HRTP_MAGIC,0,RTP_TIMEOUT)) or 1)
                                \
                or
#define ACCEPT_RTP2
                                        \
                (get <dst,dport,src,HRTP_MAGIC,ip_p> from pending to sr10,
        \
```

```
                record <rconn;0,sr11 | _UDP_ESTABLISHED,
                        \

        SPOOF_CACHE_EMPTY|TRACKED_TRANS(sr12)|RECORD_SRC(0xa8)
        \
                                                        @RTP_TIMEOUT> in
connections,                            \
                record <udprconn;DUP_KEY(sr10),sr11 | _UDP_ESTABLISHED,
                \

        SPOOF_CACHE_EMPTY|TRACKED_TRANS(sr12)|RECORD_SRC(0xa8)
        \
                                                        @RTP_TIMEOUT> in
connections,                            \
                delete <dst,dport,src,HRTP_MAGIC,ip_p> from pending,
        \
                set sr9 2,
                                                \
                (ENTRY_TRACKED(sr12),
                                        \
                RACCOUNT_MATCH(HRTP_MAGIC,0,RTP_TIMEOUT)) or 1),
\
                accept_udp_noncrypt(sr11,sr9)

#define ACCEPT_T120                                     \
      tcp, first,                                                                   \
      TABLE_NOT_EMPTY(pending),
                \
      (get <src,HRTP_MAGIC,dst,dport,ip_p> from pending to sr10,        \
         record <conn;DUP_KEY(sr10),ENTRY_TYPE(sr11),
         \
                RECORD_SRC(0xa8)|IS_ACCEPTED_A|                 \
                SPOOF_CACHE_EMPTY|TRACKED_TRANS(sr12)           \
                        @TCP_TIMEOUT> in connections,   \
         (ENTRY_TRACKED(sr12),                          \
            ACCOUNT_MATCH(HRTP_MAGIC,0,RTP_TIMEOUT)) or 1),
            \
      accept_tcp_noncrypt(sr11,1)



#define ACCEPT_RTP ACCEPT_RTP1 ACCEPT_RTP2

#define h323_prolog_new RECORD_H245_NEW; ACCEPT_H245_NEW;
RECORD_RTP_NEW;
#define h323_prematch_new ACCEPT_RTP; ACCEPT_T120;
```

```
/* old implementation */
deffunc GET_H245_PORT(WhatDetected, WhatPort) {
        (sr2 & WhatDetected) =      WhatDetected,
        set sr13 (call KFUNC_GET_RTP<WhatPort>),
        set sr1 ((sr13.[0:1]) << 8) + sr13.[1:1],
        NOTSERVER_UDP_PORT(sr1) or reject
};

deffunc SAVE_H245_PORT(port,dir,key,type,flags) {
        (
                (dir = H323DIRECTION_OUT,
                        record <src,port,dst,HRTP_MAGIC,PROTO_udp;key,type,flags

        @RTP_TIMEOUT> in pending,
                        (ENTRY_TRACKED(flags),
                         record
<0,src,port,dst,HRTP_MAGIC,PROTO_udp;sr3,sr4,sr5,sr6,sr7

        @RTP_TIMEOUT> in tracked)
                        or 1
                )
                or
                (
                        record <dst,HRTP_MAGIC,src,port,PROTO_udp;key,type,flags

        @RTP_TIMEOUT> in pending,
                        (ENTRY_TRACKED(flags),
                         record
<0,dst,HRTP_MAGIC,src,port,PROTO_udp;sr3,sr4,sr5,sr6,sr7

        @RTP_TIMEOUT> in tracked)
                        or 1
                )
        )
};


/* KFUNC_FOLLOW_H225 returns a number which caneither consist of the two
 * desired ports of the setup message - sport in the upper 16 bits, dport
 * in the lower 16 bits.
 * If the packet is a Connect message, we only have lower 16 bits.
 */
```

```
#define SPLIT_PORT(offset)
                \
        (set sr4 (offset >> 16),
        \
         set sr3 (offset & 0x0000ffff))


#ifndef NO_XLATION

/* This is for the Connect message - the port command. */
#define TRANSLATE_H225(port)
                \
        (call KFUNC_XLATE_ANTICIPATE <dst,0,0,port,PROTO_tcp,
        \
                                BUILD_CONT_REV(0,1), RTP_TIMEOUT,0,sr3,sr3 - 4>)

/* This is for the Setup message - we need to change the identifiers
 * of the connection inside the data of the packet.
 */
#define CHANGE_SETUP
                        \
        (call KFUNC_XLATE_ANTICIPATE <conn,
                        \
                                BUILD_CONT_REV(1,0), RTP_TIMEOUT,0,sr3,sr3 - 4>,
        \
         call KFUNC_XLATE_ANTICIPATE <conn,
                        \
                                BUILD_CONT_REV(2,0), RTP_TIMEOUT,0,sr4,sr4 - 4>)

#define TRANSLATE_FIRST(port)
                \
        (set sr3 sr13 - 4,
                        \
         ((r_cdir=2,call KFUNC_XLATE_ANTICIPATE <dst,0,0,port,PROTO_udp,\
                                BUILD_CONT_REV(0,1), RTP_TIMEOUT,0,sr13,sr13 -
4>) or  \
         (call KFUNC_XLATE_ANTICIPATE <src,port,dst,0,PROTO_udp,
        \
                                BUILD_CONT_REV(0,0), RTP_TIMEOUT,0,sr13,sr13 -
4>)))

#define TRANSLATE_SECOND(port)
                        \
        (set sr3 sr13 + 3,
                        \
```

```
        set sr4 sr3 + 4,
                \
   ((r_cdir=2,call KFUNC_XLATE_ANTICIPATE <dst,0,0,port,PROTO_udp,     \
                                BUILD_CONT_REV(0,1), RTP_TIMEOUT,0,sr13 +
7,sr13 + 3>) or          \
         (call KFUNC_XLATE_ANTICIPATE <src,port,dst,0,PROTO_udp,
        \
                                BUILD_CONT_REV(0,0), RTP_TIMEOUT,0,sr13 +
7,sr13 + 3>)))
#else
#define TRANSLATE_H225(port)    1
#define CHANGE_SETUP                    1
#define TRANSLATE_FIRST(port)  1
#define TRANSLATE_SECOND(port)        1
#endif



/*
 * intercept H.225 CONNECT/SETUP message, and record the
 * H.245 dynamic connection that we should accept.
 * If the H.225 connection started from A to B, We are only
 * interested in tracking packets going from B to A, since the
 * CONNECT/SETUP message is in that direction.
 * For first packet of H.225 (B->A) we init the align machine.
 * For every packet of H.225 (B->A) we call the align machine.
 * KFUNC_FOLLOW_H225 return values:
 * 0 = Bad Packet
 * 1 = H.225 connect not found in packet
 * not 0, not 1 - the offset of the port in CONNECT/SETUP message
 */


#define FIND_SETUP
                                                \
        (
                                                        \
                tcp, first, dport = H225_PORT, direction = H323DIRECTION_OUT,
        \
                call KFUNC_INIT_H225<direction>
                                \
        )

#define RECORD_H245_1
                                \
```

```
        (
                                                        \
                (r_cdir = 2, sport = H225_PORT) or (r_cdir =1,dport = H225_PORT),
        \
                tcp, ((syn, ack, direction = H323DIRECTION_IN,
                \
                                call KFUNC_INIT_H225<direction> ) or 1),
                \
                set sr1 call KFUNC_FOLLOW_H225<direction>,
                        \
                (sr1 != H_BAD_PACKET) or reject,
            \
                (
                                                        \
                        sr1 != H_NOTHING_FOUND,
                                \
                        SPLIT_PORT(sr1),                                        \
                        set sr2 (((sr3.[0:1]) << 8) + sr3.[1:1]),
#define RECORD_H245_2
                            \
                        NOTSERVER_TCP_PORT(sr2) or reject,
                                \
                        ((sr4=0, record
<dst,H245_MAGIC,src,sr2,ip_p;DUP_KEY(r_ckey),       \
                                                        r_ctype,
r_cflags@H245_TIMEOUT> in pending,    \
                        TRANSLATE_H225(sr2)) or CHANGE_SETUP),
                                \
                        (ENTRY_TRACKED(r_cflags),
                                    \
                        record
<0,dst,H245_MAGIC,src,sr2,ip_p;rconn@H245_TIMEOUT>                  \

                                in tracked)     \
                        or 1
                                                \
                ) or 1,
                                                \
                accept_fwz_as_clear(r_ctype)
                        \
        )

#define RECORD_H245 FIND_SETUP; RECORD_H245_1 RECORD_H245_2


/*
```

```
 * on first H.245 packet move the connection to connections table
 * with key and type of H.225 (they are in the same direction).
 * Keep the H.245 connection in pending table - to intercept the
 * ESTABLISH_LOGICAL_CHANNEL_ACK commands.
 * Initialize the align machine that will look for ESTABLISH_LOGICAL_CHANNEL
 * req/ack
 * This packet is accepted by  IS_ACCEPTED_A  if clear, or connection table
 *  if encrypted
 */

#define ACCEPT_H245_FWD1
                                        \
                tcp, first,
                                                \
                TABLE_NOT_EMPTY(pending),
                                        \
                get <src,H245_MAGIC,dst,dport,ip_p> from pending to sr10,
        \
                NOT_TCP_FASTMODE_PORT(sport,0) or reject,
                        \
                record
<conn;DUP_KEY(sr10),ENTRY_TYPE(sr11),SPOOF_CACHE_EMPTY |          \

IS_ACCEPTED_A|TRACKED_TRANS(sr12)|MORE_INSPECTION|RECORD_SRC(0
x88)\

        @TCP_TIMEOUT> in connections, \
                (ENTRY_TRACKED(sr12),
#define ACCEPT_H245_FWD2
                                        \
                ACCOUNT_MATCH(H245_MAGIC,0,TCP_TIMEOUT)) or 1,
                                \
                (direction=H323DIRECTION_OUT,call
KFUNC_INIT_H245<direction>) or 1,        \
            update_tcp_flags(sr11,1)

#define ACCEPT_H245_FWD ACCEPT_H245_FWD1 ACCEPT_H245_FWD2

#define ACCEPT_H245_BAK
                                                \
                tcp, syn, ack,
                                        \
                TABLE_NOT_EMPTY(pending),
                                        \
                get <dst,H245_MAGIC,src,sport,ip_p> from pending to sr10,          \
```

```
                (direction=H323DIRECTION_IN,call KFUNC_INIT_H245<direction>)
or 1


#define ACCEPT_H245 ACCEPT_H245_FWD; ACCEPT_H245_BAK

/* RECORD_RTP macro broken to A few macros
 * intercept the ESTABLISH_LOGICAL_CHANNEL_ACK command (on the H.245
 * connection).
 * Record both the RTP (audio/video) and RTCP (control) (+1) connections.
 * The align machine looks for ESTABLISH_LOGICAL_CHANNEL_ACK in both
direction.
 * KFUNC_FOLLOW_H245 return codes:
 * 0 Bad packet
 * 1 Nothing found in packet
 * EST_LOG_CHAN_ACK (2) - Establish logical channel ack found in packet
 * EST_LOG_CHAN (4) - Establish logical channel found in packet
 * the last 2 may arrive together!
 * KFUNC_GET_RTP first argument:
 * 0 - get offset of RTCP port in open logical channel msg
 * 1 - get offset of RTCP port in open logical chan ack msg
 * 2 - get offset of RTP port in open logical chan ack msg
 */

#define FOLLOW_H245
                                                \
        (
                                                        \
                set sr2 (call KFUNC_FOLLOW_H245<direction>),
                \
                (sr2 != H_BAD_PACKET) or reject,
                        \
                sr2 != H_NOTHING_FOUND,                         \
                (ENTRY_TYPE(sr11) = CONN_TCP, set sr9 CONN_UDP)
                                \
                  or
                                                \
                (set sr9 ENTRY_TYPE(sr11))
                        \
        )


#define GET_SAVE_PORT_1
                                                \
(               GET_H245_PORT(H_EST_LOG_CHAN_ACK_2, H_ACK_RTP_2),
                                \
```

```
                SAVE_H245_PORT(sr1,sr8,DUP_KEY(sr10),sr9,sr12),
                        \
                TRANSLATE_FIRST(sr1),
                                \
                SAVE_H245_PORT(sr1+1,sr8,DUP_KEY(sr10),sr9,sr12),
                \
                TRANSLATE_SECOND(sr1+1)
                                        \
)


#define GET_SAVE_PORT_2                                         \
(               GET_H245_PORT(H_EST_LOG_CHAN, H_REQ_RTCP),
                                \
                SAVE_H245_PORT(sr1,sr8,DUP_KEY(sr10),sr9,sr12),
                        \
                TRANSLATE_FIRST(sr1),
                                \
                GET_H245_PORT(H_EST_LOG_CHAN_2, H_REQ_RTCP_2),
                        \
                SAVE_H245_PORT(sr1,sr8,DUP_KEY(sr10),sr9,sr12),
                        \
                TRANSLATE_FIRST(sr1)
                                \
)


#define RECORD_RTP1
                \
        (
                                                        \
                r_cdir, tcp, not_first, TABLE_NOT_EMPTY(pending),
                \
                (
                                                        \
                        get <src,H245_MAGIC,dst,dport,ip_p> from pending to sr10,
        \
                        set sr8 1
                                        \
                ) or (
                                        \
                        get <dst,H245_MAGIC,src,sport,ip_p> from pending to sr10,
        \
                        set sr8 2
                                        \
```

```
                ),
#define RECORD_RTP2
                                                \
                (
                                                \
                        FOLLOW_H245,
                \
                        (ENTRY_TRACKED(sr12),
                                        \
                        (sr8 = 1, get <0,conn> from tracked to sr3)
                \
                         or
                                        \
                        (get <0,rconn> from tracked to sr3)) or 1,
                \
                   (
                                        \
                                (                                       \
                                        GET_H245_PORT(H_EST_LOG_CHAN_ACK,
H_ACK_RTP),                             \

        SAVE_H245_PORT(sr1,sr8,DUP_KEY(sr10),sr9,sr12),                 \
                                        TRANSLATE_FIRST(sr1),
#define RECORD_RTP3
                                        \

        SAVE_H245_PORT(sr1+1,sr8,DUP_KEY(sr10),sr9,sr12),       \
                                        TRANSLATE_SECOND(sr1+1),
                                        \
                                        GET_SAVE_PORT_1,
                                                \
                                        0                               \
                                ) or (                          \
                                        GET_SAVE_PORT_2
                                                \
                                )
                                                \
                        )
                                                \
                ) or 1,
                                        \
                accept_fwz_as_clear(r_ctype)
                        \
        )

#define RECORD_RTP RECORD_RTP1 RECORD_RTP2 RECORD_RTP3
```

```
#define h323_prolog RECORD_H245; ACCEPT_H245; RECORD_RTP;
#define h323_prematch ACCEPT_RTP;

#else /* NO_H323 */
#define h323_prolog_new 1;
#define h323_prematch_new 1;
#define h323_prolog 1;
#define h323_prematch 1;
#endif
```

FWUI head definition

```
#ifndef __solaris2
#define __solaris2
#endif
#ifndef __fwui_head__
#define __fwui_head__

#ifdef FWGCC
#define CONCAT(a,b) a##b
#else

#ifdef FWCPP
#define CONCAT(a,b) a#b
#endif

#ifdef __freebsd
#define CONCAT(a,b) a#b
#endif

#ifdef __solaris2_i386
#define CONCAT(a,b) a#b
#endif

#ifdef __solaris2
#define CONCAT(a,b) a#b
#endif

#ifdef __hpux
#define CONCAT(a,b) a/**/b
#endif

#ifdef __sunos4
#define CONCAT(a,b) a/**/b
#endif

#ifdef __rs6000
#define CONCAT(a,b) a#b
#endif

#ifdef __unixware
#define CONCAT(a,b) a#b
#endif

#ifdef __WIN32
#define CONCAT(a,b) a#b
```

```
#endif
#endif

#define r_ckey_ctype_cflags  r_ckey
#define r_xlate_pool sr15

#define src ip_src
#define dst ip_dst
#define sport th_sport
#define dport th_dport

#define STATEFUL_ICMP
/*
 * Uncomment the following line to enable stateful ICMP violations logging
 */
/* #define STATEFUL_ICMP_LOG */

/*
 * Uncomment the following define to activate SSO
 */
//#define HAS_SSO 1

/*
 * Uncomment the following line to enable TCP Non-SYN packet to go through
 * the rule-base.
 */
/*#define ALLOW_NON_SYN_RULEBASE_MATCH */

/*
 * Comment the following line to disable logging of TCP Non-SYN packets dropped
 * because they are not alowed to go through the rule-base
 */
#define NON_SYN_RULEBASE_MATCH_LOG

#include "tcpip.def"
#include "fwconn.h"
#include "traps.h"
#include "kerntabs.h"
#include "table.def"
#include "../conf/xlate.conf"
#include "formats.def"
#include "traps.def"


#define LOG_NOALERT
```

```
#ifdef LOG_ESTABLISHED_TCP
#define LOG_TCP 1
#else
#define LOG_TCP ((ip_p = 6, first) or ip_p != 6)
#endif

#ifdef LOG_IMPLIED_RULES
#define IMPLIED_LOG LOG(long, LOG_NOALERT, 0)
#else
#define IMPLIED_LOG (1)
#endif

#define LOG_ICMP (ip_p = 1)

/*
 *   LOG(name, alert, rule) logs current packet using format name with optional
 * alert. The rule argument is a number, logged under the 'rule' log field.  If
 * LOG_TIMEOUT is not zero the packet is logged only if it is not in the logged
 * table.
 */
#if LOG_TIMEOUT > 0
deffunc LOG_test(rule) {
                (LOG_ICMP, ((<ip_p,src,dst,rule> in logged), set sr1 1) or
                     ((record <ip_p,src,dst,rule> in logged),set sr1 2))
                or
                ( <dst,ip_p,dport> in rpc_serv,
                        ((<ip_p,src,dst,sport,dport,rule> in logged),set sr1 1) or
                        (record <ip_p,src,dst,sport,dport,rule> in logged,set sr1 3))
                or
                (LOG_TCP,
                        ((<ip_p,src,dst,sport,dport,rule> in logged),set sr1 1) or
                        (record <ip_p,src,dst,sport,dport,rule> in logged,set sr1 4))
};

#define LOG(name,alert,rule)
                        \
        (
                                                        \
                (LOG_test(rule),
                                        \
                 (
                                                        \
                        (sr1 = 2, set sr1 rule, log alert CONCAT(icmp_,name)) or
        \
                        (sr1 = 3, set sr1 rule, log alert CONCAT(rpc_,name)) or
        \
```

```
                        (sr1 = 4, set sr1 rule, log alert name)
                \
                  )
                                                        \
                ) or 1
                                                \
        )

#else
#define LOG(name,alert,rule)
                        \
                (set sr1 rule,
                                        \
                        (LOG_ICMP, log alert CONCAT(icmp_,name))
                        \
          or
                                                \
                        ( <dst,ip_p,dport> in rpc_serv, rpc_serv[dst,ip_p,dport] is cb_prog,
        \
                         log alert CONCAT(rpc_,name))
                                \
          or
                                                \
                        (LOG_TCP, log alert name)
                                \
                or 1)
#endif


#define conn src,sport,dst,dport,ip_p
#define udpconn src,sport,dst,0,ip_p

deffunc ACCOUNT_RECORD(rule) {
        (
                (
                        (<conn> in tracked)
                        or
                        (record <conn;date+tod,0,0,rule,0,ifid,date+tod> in tracked)
                ),
                get <conn> from connections to r_ckey_ctype_cflags,
                set r_cflags (r_cflags | IS_TRACKED_UNK),
                modify <conn;r_ckey,r_ctype,r_cflags> in connections,
                (
                        ip_p = udp,
                        record <0,udpconn;conn@UDP_TIMEOUT> in tracked,
              get <udpconn> from connections to r_ckey_ctype_cflags,
```

```
                    set r_cflags (r_cflags | IS_TRACKED_TRANS_UNK),
                        modify <udpconn;r_ckey,r_ctype,r_cflags> in connections
                ) or 1
        )
};

#undef conn
#undef udpconn


/*
 *  ACCOUNT macro first calls LOG, then records the connection in
 *  the 'tracked' table.
 *  Notice that in the ACCOUNT macro the rule argument is passed with
 *  the most bit set to signify there is supposed to be  accounting .
 */
#define ACCOUNT(name,alert,rule)                                    \
   (                                                    \
      LOG(name,alert,(rule | 0x80000000)),                          \
      (                                                 \
         (tcp or udp), ACCOUNT_RECORD(rule)
                \
      ) or 1                                            \
   )


/*
 * Balancing (Logical Server) Macro for http and auth logical servers
 */
#define IS_LOGICAL_REQUEST(rule)
                                    \
        (
                                                            \
                <conn, rule> in logical_requests
                \
        )

/*
 * Balancing (Logical Server) Macro for other logical servers.
 * continue established conn after reload
 */
#define IS_NEW_OR_OLD_LOGICAL_REQUEST(rule)
                                    \
        (
                                                            \
```

```
                (
        \
                        <conn, rule> in logical_requests
                        \
                ) or (
                \
                        tcp, established,
        \
        <conn> in old_connections
        \
                )
        \
        )

/*
 * Balancing (Logical Server) cache Macro
 * sr10 - new dst
 * sr11 - new dport
 * sr12 - if url-redirection then holds the fold port, else holds 0
 */
#define LOGICAL_CACHE(rule)
                \
        (
                                                        \
                get <src,dst,dport,rule> from LOGICAL_CACHE_TABLE to sr10,
        \
           (
                                                \
             (
                                                \
                        sr12 = 0,                                       \
                        call KFUNC_XLATE_FOLD<sr10,sr11,2>,
        \
                    record <conn,rule> in logical_requests
                \
             )  or (
                                        \
                        sr12,                                           \
                        call KFUNC_XLATE_FOLD<ifaddr,sr12,2>,
        \
                    record <src,sport;sr10> in logical_requests
                \
             )
                                                \
           )
                                                \
```

```
        )


/*
 * Balancing (Logical Server) Macro
 */
#define LOGICAL_INVOKE(rule)
                                \
        (
                                                        \
                (
                                                        \
                        <conn> in logical_requests, drop
                        \
                ) or (
                                                \
                        record <conn> in logical_requests,
                        \
                        set sr1 rule, set sr2(call KFUNC_LOAD_BALANCE                   \
                           <src, sport, dst, dport, ip_p, sr1, packetid>),              \
                        (sr2 or drop), ((sr2 = 2, hold) or (sr2 != 2, 1))
        \
                )
                                                        \
        )


/*
 * Balancing (Logical Server) Macro with caching
 */
#define LOGICAL_INVOKE_CACHED(rule)
                                        \
        (
                                                        \
                (
                                                        \
                        LOGICAL_CACHE(rule)
                                \
                ) or (
                                                \
                        LOGICAL_INVOKE(rule)
                                        \
                )
                                                        \
        )
```

```
#define LOGICAL_FOLD(naddr, nport)                                  \
        (
                                                                    \
                call KFUNC_XLATE_FOLD <naddr, nport, 2>,                    \
                record <src, sport, naddr, nport, ip_p> in accepted
        \
   )

redirect_requests = dynamic expires 10;

#define SRV_REDIRECT(oldport, ndst, nport) \
       ( \
                direction = 0, tcp, dport = oldport, \
                call KFUNC_XLATE_FOLD <ndst, nport, 2>, \
                record <conn> in redirect_requests \
        ) or ( \
                direction = 1, <conn> in redirect_requests \
        )

/*
 * TRAP macros:
 *
 * TRAP(name, rule) traps the daemon with the 'name' trap format.
 *
 * TRAP_NP(name, rule) does the same as TRAP() but does not consider the source
 * and destination ports as part of the index to the 'trapped' table.
 *
 */
#define TRAP(name,rule)
                                        \
                (<conn,rule> in trapped
                                        \
            or
                                                \
                (record <conn,rule> in trapped, set sr1 rule, log name))

#define TRAP_NP(name,rule)
                                        \
                (<ip_p,src,dst,rule> in trapped
                                        \
            or
                                                \
                (record <ip_p,src,dst,rule> in trapped, set sr1 rule, log name))

#define TRAP_TO(name,rule,timeout)
                                        \
```

```
                (<conn,rule> in trapped
                                        \
            or
                                                \
                (record <conn,rule@timeout> in trapped, set sr1 rule, log name))

#define TRAP_FLG(name,rule,flags)
                                \
                (<conn,rule | flags> in trapped
                                \
            or
                                                \
                (record <conn,rule | flags> in trapped, set sr1 rule, log name))

/*
 * Definitions for the needruleinfo service attributes.
 */
#define set_rule(val) (set sr1 (val))
#define set_log(val)  (set sr2 (val))


#include "base.def"

/*
 *   Do not delete the following definitions !  We need them due to the way our
 * code generator works.
 */
#define targets hosts
#define gateways
#define logics
#define switchs
#define routers
#define broadcasts
#define blackboxs

// the start_rule_code is a marco which runs before every generated rule
#if defined(FTP_PORT_RESTR)
#define start_rule_code(rn) (set r_entry MATCH_BY_PROTOCOL, set sr3 0, delete
<conn> from ftp_restrictions)
#else
#define start_rule_code(rn) (set r_entry MATCH_BY_PROTOCOL, set sr3 0)
#endif
#define SRV_tcp(name,addr) define name { dport = addr };
#define SRV_udp(name,addr) define name { dport = addr };
#define SRV_rpc(name,addr) define name { rpc_prog(addr) };
#define SRV_icmp(name,expr) define name { expr };
```

```
#define SRV_other(name,expr) define name { expr, set sr3 MORE_INSPECTION};
#define SRV_tcp_range(name,sl,sh,dl,dh) define name \
  { sport >= sl , sport <= sh , dport >= dl, dport <= dh };
#define SRV_udp_range(name,sl,sh,dl,dh) define name \
  { sport >= sl , sport <= sh , dport >= dl, dport <= dh };

#define ADDR_domain(name, addr)
#define ADDR_host(name,addr) define name { addr };
#define ADDR_net(name,addr,nm) define name { addr }; define
CONCAT(name,_netmask) { nm };
#define ADDR_network(name,addr,nm) define name { addr }; define
CONCAT(name,_netmask) { nm };
#define ADDR_gateway(name,addr) define name { addr };
#define ADDR_gateway_cluster(name,addr) define name { addr };
#define ADDR_switch(name,addr) define name { addr };
#define ADDR_blackbox(name,addr) define name { addr };
#define ADDR_router(name,addr) define name { addr };
#define ADDR_hostif(name,addr) define name { addr };
#define ADDR_logical(name,addr) define name { addr };
#define ADDR_machines_range(name, start, end) define name {start} ; define
CONCAT(name,_start) { start };          define CONCAT(name,_end) {end};
#define ADDR_valid(name,addr) define CONCAT(name,_valid) {addr};

#define net_notin(addr,net,mask) ((addr & mask) != net)
#define net_in(addr,net,mask) ((addr & mask) = net)

#define MAKE_ALERT(alert_tab, alert) alert_tab = format alert { };

#define PORTS_tcp(name, start, end) define CONCAT(name,_start) {start};
define CONCAT(name,_end) {end};
#define PORTS_udp(name, start, end) define CONCAT(name,_start) {start};
define CONCAT(name,_end) {end};
#define PORTS_ports_range(name, start, end) define CONCAT(name,_start) {start};
define CONCAT(name,_end) {end};

#define RANGE_MACHINE(name)  name,name
#define RANGE_NETWORK(name) name,name | ~ CONCAT(name,_netmask)
#define RANGE_MACHINES_RANGE(name)
        CONCAT(name,_start),CONCAT(name,_end)
#define RANGE_TCP(name)         CONCAT(name,_start),CONCAT(name,_end)
#define RANGE_UDP(name)         CONCAT(name,_start),CONCAT(name,_end)
#define RANGE_PORTS_RANGE(name)
        CONCAT(name,_start),CONCAT(name,_end)

#define VALID_ADDR(name)        CONCAT(name,_valid)
```

```
#define VALID_ADDR_RANGE_MACHINE(name)
        CONCAT(name,_valid),CONCAT(name,_valid)
#define VALID_ADDR_RANGE_NETWORK(name)        CONCAT(name,_valid),\
                                        CONCAT(name,_valid) + (name | ~
CONCAT(name,_netmask)) - name
#define VALID_ADDR_RANGE_MACHINE_RANGE(name)
        CONCAT(name,_valid),\
                                        CONCAT(name,_valid) + CONCAT(name,_end) -
CONCAT(name,_start)

#define TABLE_INDEX(num1,num2)        (((num1)<<16)|(num2))

#include "auth.def"
#include "dup.def"
```

xtreme definitions

VXTREME (Web Theater)

```
/*
 * The client opens a TCP connection with the server on port 12468 only.
 * The media stream uses RTP so it binds 2 ports for each stream, the port#
 *  that is given in the TCP packet and port#+1.
 * The format of XTREME packet is: each message is preceded by 2 bytes for
 * the length.
 *
 * ---------------------------------------------------------------------
 * msgid|ver|#streams|cmdseq#|*CMD*|streamid|#Attrib|Attr|opt|len|value|cmdseq
 * ---------------------------------------------------------------------
 * (42) (44)         (46)         (48)           (50)     (52) (54)
 * 16     8    8          8    8      8        8       8   8   16  32x
 * CMD:    if CMD=16 this is a connect request from the client.
 *         if CMD=0 this is a success reply from the server.
 * #attrib:how many attributes follow this command. for CmdConnect this
 *         is usually 9 for UDP connection, and 10 for TCP.
 * attrib: attrib=1, (usually the first attrib)means this is the port number
 *         for the future UDP/TCP connection.
 *           attrib=74 means that the future connection will be TCP only.
 * Download a win95/nt client from http://www.vxtreme.com .
 */

#define XTREME      12468

#define XTREME_clientheader                                    \
        [TCPDATA+4:4,b]=0x01010010
                                    /* version#1 & 1stream &connection req.*/
#define XTREME_serverheader                                            \
        [TCPDATA+4:4,b]=0x01010000
   /*version#1 & 1stream  & success reply*/

#define XTREME_NumOfAttrib  [TCPDATA+9:1]

#define XTREME_timeout 180

#define XTREMEportlen
        \
        (([TCPDATA+12:1] << 8) + [TCPDATA+13:1])

#define XTREME_port
        \
```

```
        (([TCPDATA+16:1] << 8) + [TCPDATA+17:1])
    /* the information regarding the future UDP or TCP conn */

#define XTREME_port_offset
        \
        (TCPDATA+14+2)
        /* from the beginning of the packet */

#define XTREME_msgId [TCPDATA+2:2]
        /* each stream has a unique num. */

#define XTREME_make4mul(num)
                                \
  ((num&3 =3,
                                        \
         set sr1 (num+1))
                                \
                    or
                                        \
        ((num+1)&3 =3,
                                        \
         set sr1 (num+2))
                                \
                    or
                                        \
        ((num+2)&3 =3,
                                        \
         set sr1 (num+3))
                                \
                    or
                                        \
        ((num+3)&3 =3,
                                        \
         set sr1(num)))

#define XTREME_advance(locat)
                                \
  (XTREME_make4mul([TCPDATA+locat:2]),
                                \
        set sr2 (locat+sr1+2))


deffunc XTREME_proto() {
  (XTREME_make4mul(XTREMEportlen),
```

```
set sr2 (14+sr1),
[TCPDATA+sr2:1,b]=74                /*checking attrib #2 */
                or
(XTREME_advance(sr2+2),
 [TCPDATA+sr2:1,b]=74)              /* checking attrib #3 */
                or
(XTREME_advance(sr2+2),
 [TCPDATA+sr2:1,b]=74)              /* checking attrib #4 */
                or
(XTREME_advance(sr2+2),
 [TCPDATA+sr2:1,b]=74)              /* checking attrib #5 */
                or
(XTREME_advance(sr2+2),
 [TCPDATA+sr2:1,b]=74)              /* checking attrib #6 */
                or
(XTREME_advance(sr2+2),
 [TCPDATA+sr2:1,b]=74)              /* checking attrib #7 */
                or
(XTREME_advance(sr2+2),
 [TCPDATA+sr2:1,b]=74)              /* checking attrib #8 */
                or
(XTREME_NumOfAttrib>8,
 XTREME_advance(sr2+2),
```

```
        [TCPDATA+sr2:1,b]=74)              /* checking attrib #9 */

                        or

        (XTREME_NumOfAttrib>9,

         XTREME_advance(sr2+2),

        [TCPDATA+sr2:1,b]=74)              /* checking attrib #10*/

                        or

        (XTREME_NumOfAttrib>10,

         XTREME_advance(sr2+2),

        [TCPDATA+sr2:1,b]=74))             /* checking attrib #11*/
};


deffunc XTREME_notserver(port) {
  (NOTSERVER_UDP_PORT(port) or reject,

        NOTSERVER_UDP_PORT((port)+1) or reject)
};
                        /* rtp protocol, we need to bind 2 ports*/

#define rec_fl TRACKED_TRANS(r_cflags)|SPOOF_CACHE_EMPTY

#define xtreme_record(s,sp,d,dp,pr,k,t,fl,to)
        \
        (pr=PROTO_tcp,
                                                \
         record <s,sp,d,dp,pr;k,t,fl|rec_fl @to> in connections)
        \
                                or
                                                        \
    (record <s,sp,d,0 ,pr;k,t|_UDP_ESTABLISHED,fl|rec_fl @to>
        \

                in connections,                 \
    record <s,sp,d,dp,pr;0,t|_UDP_ESTABLISHED,fl|rec_fl @to>
        \

                in connections)
```

```
#define xtreme_record_tcp_clear(sourc,srcpor,des,dstpor)
        \
        (ENTRY_TYPE(r_ctype)=CONN_TCP,
                                        \
         xtreme_record(sourc,srcpor  ,des,dstpor  ,PROTO_tcp,0,CONN_TCP,
        \
                                                IS_ACCEPTED_A|RECORD_SRC(0x90),
        TCP_START_TIMEOUT),   \
         xtreme_record(sourc,srcpor+1,des,dstpor+1,PROTO_tcp,0,CONN_TCP,
        \
                                                IS_ACCEPTED_A|RECORD_SRC(0x91),
TCP_START_TIMEOUT))

#define xtreme_record_tcp_enc(sourc,srcpor,des,dstpor)
        \
        (set sr6 REMOVE_CTRL(r_ctype, TCP_CTRLS),
                        \
         xtreme_record(sourc,srcpor,des,dstpor,PROTO_tcp,DUP_KEY(r_ckey),
        \

        sr6,RECORD_SRC(0x92),TCP_START_TIMEOUT),       \
         xtreme_record(sourc,srcpor+1,des,dstpor+1,PROTO_tcp,DUP_KEY(r_ckey),
        \

        sr6,RECORD_SRC(0x93),TCP_START_TIMEOUT))

deffunc xtreme_record_tcp(sourc,srcpor,des,dstpor) {
         (NOT_TCP_FASTMODE_PORT(srcpor,0),

                (xtreme_record_tcp_clear(sourc,srcpor,des,dstpor))

         or

                (xtreme_record_tcp_enc(sourc,srcpor,des,dstpor))

         )
};


deffunc xtreme_record_udp(sourc,srcpor,des,dstpor) {
         (((ENTRY_TYPE(r_ctype) = CONN_TCP, set sr6 CONN_UDP)

                                or
```

```
          set sr6 r_ctype),

         xtreme_record(sourc,srcpor,des,dstpor,PROTO_udp,DUP_KEY(r_ckey),
                                        sr6,MORE_INSPECTION | RECORD_SRC(0x94),
        XTREME_timeout),
         xtreme_record(sourc,srcpor+1,des,dstpor+1,PROTO_udp,DUP_KEY(r_ckey),
                                        sr6,MORE_INSPECTION | RECORD_SRC(0x95),
XTREME_timeout))
};

deffunc xtreme_account(sourc,srcpor,des,dstpor,prot) {
        ((ENTRY_TRACKED(r_cflags),

    record <0,sourc,(srcpor)  ,des,(dstpor)  ,prot; rconn @PENDING_TIMEOUT>

                                in tracked,
         record <0,sourc,(srcpor+1),des,(dstpor+1),prot; rconn @PENDING_TIMEOUT>

                        in tracked) or 1)
};

#define xtreme_anticipate_client(prtcl)
                        \
        (call KFUNC_XLATE_ANTICIPATE
                                        \
                <0,sr1,dst,0,prtcl,BUILD_CONT_REV(0,0),XTREME_timeout,0,sr2,0>,
        \
         call KFUNC_XLATE_ANTICIPATE
                                        \
                <0,sr1+1,dst,0,prtcl,BUILD_CONT_REV(0,0),XTREME_timeout>)

#define xtreme_get_client                                       \
        r_cdir = 1, dport=XTREME or dport=XTREME+1, tcp, established,
        \
        packetlen >= 80,XTREME_clientheader,
                        \
        XTREME_notserver(XTREME_port),
        \
        (XTREME_proto, set sr3 6) or set sr3 17,
                \
        set sr1 XTREME_port, set sr2 XTREME_port_offset,
                \
        xtreme_anticipate_client(sr3),
                                \
        record <rconn;0,0,sr1,XTREME_msgId,sr3> in pending,
                \
```

```
        accept_fwz_as_clear(r_ctype)

#define xtreme_anticipate_server(prtcl)
                        \
        (call KFUNC_XLATE_ANTICIPATE
                                        \

        <dst,sr3,0,sr1,prtcl,BUILD_CONT_REV(0,1),XTREME_timeout,0,sr2,0>,
        \
         call KFUNC_XLATE_ANTICIPATE
                                        \
                <dst,sr3+1,0,sr1+1,prtcl,BUILD_CONT_REV(0,1),XTREME_timeout>)

#define xtreme_get_server1                                      \
        r_cdir = 2, sport=XTREME or sport=XTREME+1, tcp,
                \
        packetlen >= 60, 1 or TABLE_NOT_EMPTY(pending),
                        \
        XTREME_serverheader,
                                        \
        XTREME_notserver(XTREME_port),
        \
        get <conn> from pending to sr1,
        \
        sr4 = XTREME_msgId,
#define xtreme_get_server2  set sr1 XTREME_port, set sr2 XTREME_port_offset,
                                        \
        xtreme_anticipate_server(sr5),
                                \
        (sr5 = PROTO_udp, xtreme_record_udp(dst,sr3,src,sr1))
        \
                        or
                                                        \
        xtreme_record_tcp(dst,sr3,src,sr1),
                        \
        xtreme_account (dst,sr3,src,sr1,sr5),
                \
        delete <conn> from pending, accept_fwz_as_clear(r_ctype)

#define xtreme_get_server xtreme_get_server1 xtreme_get_server2

#define xtreme_touch_udp udp, (r_cdir = 1, <src,sport+1,dst,dport+1,PROTO_udp> in
connections or <src,sport-1,dst,dport-1,PROTO_udp> in connections) or (r_cdir = 2,
<dst,dport-1,src,sport-1,PROTO_udp> in connections or
<dst,dport+1,dst,dport+1,PROTO_udp> in connections)
```

```
#define xtreme_code   xtreme_get_client; xtreme_get_server; xtreme_touch_udp;
```

Conclusion

Oh you're finished already?? Well then you either didn't read it properly and just checked how the story ends or you are a Checkpoint developer who accidentally deleted some sources and want to regain them again ☺

This way or the other now you know that CP firewall 1 is a very complex and robust system that if works is great but if not you better get a real guru to fix it (and beware those who just talk too much and can't handle it anyway).

If I would be designing this firewall engine I would've surely done it in a bit more different and flexible way and maybe one day I would (just as Juniper was founded to beat Cisco in all the areas routers can be beaten) but that is a story for another book..

Example 1. ICMP INSPECT SCRIPT

```
// Stateful icmp v1.3 William D. Burns (shadow@netscape.com)
// (c) Copyright 1997,1998,1999 Netscape Communications
// first released August 17, 1997.  Last updated: August 3, 1999
//
//ICMP INSPECT SCRIPT
//
//  internal testing builds:
//  -- v1.1 added icmp_ip_seq and icmp_ip_id checking
//  -- v1.2 first released version to check ping
//  -- v1.2a removed ip_ttl < 30 check for ping
//  -- v1.3 (internal) added tracert support
//  -- v1.4 tracks unix traceroute, microsoft tracert, and ping
//  -- v1.5 correctly delete spent records from tables
//
//  release builds:
//  -- v1.1 name change from ns_ping15a.def to ns_icmp11.def
//  -- v1.1 found a new bug in INSPECT.
//     + noticed equality tests did not always work
//     + and Checkpoint always used an different syntax than I
//     + to use less ()s you should test tables as
//     + 'a_table[a,b] = value'   instead of
//     + 'value = a_table[a,b]'  because this doesnt work
//     + you have to say '(a_table[a,b] = value)'
//
//  -- v1.2
//     + incorporated minor suggestions from Checkpoint
//     + implemented UNIX traceroute reply packet acceptance
//     + first release to public
//
//  -- v1.3
//     + FW-1 v4.0 added a new icmp_sequence_number define which caused
//       my code to fail to compile.  No longer defining it here.
//     + fixed bug so a host can ping and traceroute to same target
//
// ==============================================
//       Variable definition section
// redundant in FW-1 v4:
// #define icmp_icmp_seq [ 54 : 2, b ]
// ==============================================
// where icmp sequence number is
#define icmp_ip_seq [ 24 : 2, b ]
#define Timeout 5
ping_table     = dynamic {} expires Timeout;
ms_trt_table   = dynamic {} expires Timeout;
unix_trt_table = dynamic {} expires Timeout;

// ==============================================
//         Packet definition section
// ==============================================

// ==============================================
// What a ping request looks like
//
#define is_ping_request (                      \
```

```
  icmp, icmp_type=ICMP_ECHO                  \
)

// ================================================
// Per the ping of death and checkpoint web pages
// this will disallow fragmented packets from
// coming in
//
#define not_fragmented (                        \
  ((ip_off & 0x2000) = 0)                       \
)

// ================================================
// What a proper ping reply packet looks like
//
#define is_ping_reply (                                 \
  icmp, icmp_type=ICMP_ECHOREPLY             \
)

// ================================================
// What a proper ICMP Time exceeded (type 11)
//  looks like -- will come from routers in path
//
#define is_timxceed_reply (                  \
  icmp, icmp_type=ICMP_TIMXCEED                         \
)

// ================================================
// What a unix traceroute request looks like
//
#define is_traceroute_request (                 \
  udp, uh_dport > 33000, ip_ttl < 30          \
)

// ================================================
// What a unix traceroute reply looks like
//
#define is_dstunreach_reply (                   \
  icmp, icmp_type = ICMP_UNREACH              \
)

// ================================================
// Begin tracking and decision-making section
// ================================================

// ================================================
// ICMP echo request
//
// if it looks like an echo packet,
//   it is either a real ping packet
//     - record in ping_table
//     - replies will come from target
//   or it is a microsoft tracert packet
//     - record in ms_trt_table
//     - replies could come from target or
//       routers along the way
//
```

```
#define ping_request_accept (                   \
  is_ping_request, not_fragmented,            \
  record <src,icmp_ip_id; icmp_ip_seq> in ms_trt_table, \
  record <src,dst,icmp_ip_id; icmp_ip_seq> in ping_table     \
)

// ==============================================
// Got: ICMP echo reply
//
// accept the packet iff it corresponds to an
//   echo packet that we recorded
//
#define ping_reply_intercept (                  \
  is_ping_reply, not_fragmented,                \
  accept (                                      \
    icmp_ip_seq = ping_table [dst,src,icmp_ip_id], \
    delete <dst,icmp_ip_id> from ms_trt_table,       \
    delete <dst,src,icmp_ip_id> from ping_table      \
  or (log icmp_long, drop)                      \
  )                                             \
)

// ==============================================
// Got: ICMP time exceeded
//
// accept the packet iff it matches an echo packet
//  that we recorded
// Note that Ori wants us to use icmp_icmp_seq check for ms_trt_table
//
#define timxceed_reply_intercept (            \
  is_timxceed_reply, not_fragmented,          \
  accept (                                    \
   (icmp_ip_seq = ms_trt_table [dst,icmp_ip_id] \
     or                                       \
    icmp_uh_dport = unix_trt_table [dst,icmp_uh_sport]), \
    delete <dst,icmp_ip_id> from ms_trt_table,       \
    delete <dst,icmp_uh_sport> from unix_trt_table \
  or (log icmp_long, drop)                    \
  )                                           \
)

// ==============================================
// Begin UNIX traceroute section
// ==============================================
//
// Track any packets that look like "traceroute"
// packets and record them in the unix_trt_table.
//
// Responses will be similar to those for
// microsoft traceroute (icmp type 11) but will
// include icmp type 3
//

// ==============================================
// Got: UNIX Traceroute Request packet
//
// Per the RFC, the udp source and destination
```

```
// ports MUST be in the reply packet.
// Replies will be tracked against this.
//
#define traceroute_request_accept (           \
  is_traceroute_request,                      \
  record <src,uh_sport;uh_dport> in unix_trt_table   \
)

// ================================================
// Got: ICMP Destination Unreachable packet
//
// Accept the packet iff it matches a traceroute
// request that we recorded
//
#define dstunreach_reply_intercept (          \
  is_dstunreach_reply, not_fragmented,        \
  accept (                                    \
   icmp_uh_dport = unix_trt_table[dst,icmp_uh_sport],\
    delete <dst,icmp_uh_sport> from unix_trt_table \
  or (log icmp_long, drop)                    \
  )                                           \
)

// ================================================
// ================================================
// Begin ICMP REPLY checking section
// ================================================
// ================================================
// only three ICMP packets we want
//   - echo reply or
//   - time exceeded reply or
//   - destination unreachable reply
//
#define ns_icmp_reply_intercept (             \
  ping_reply_intercept or                     \
  timxceed_reply_intercept or                 \
  dstunreach_reply_intercept                  \
)


//This is what you put in the prologue section
//  and what gets executed on a match
//
#define ns_icmp_code {                        \
        ns_icmp_reply_intercept;              \
}
```