



Snort 初探

Aphyr Lee

aphyr@www.elites.org

2004.11.20

Outline

- ✦ How to IDSs detect intrusions
- ✦ Snort's Inner Workings
- ✦ Playing by the Rules
- ✦ Conclusion

How to IDSs detect intrusions (1/6)

☀ Any way they can

- Specialized trick for BackOrifice
 - Magic string: `*!*QWTY?`
 - Random generator - `((holdrand = holdrand * 214013L + 2531011L) >> 16) & 0x7fff`
 - Bruce force decryption
- Snort - `spp_bo.c/spp_bo.h` preprocessor

☀ Pattern-match

- Searching network traffic for distinctive patterns
 - Alert tcp any any -> any any (msg: "RPC EXPLOIT startdx"; content: "/bin||c74604|sh"; sid: 600;)
 - Alert tcp any any -> any any (msg: "RPC EXPLOIT startdx", content: "/bin/|c74604|sh"; sid: 1281;)
- Snort – `sp_pattern_match` detection-plugins

Ref: [How ISS RealSecure Network Sensor 7.0 Detects Intrusions](#)

How to IDSs detect intrusions (2/6)

☀ Reassembly

- Data could span more than one packets
- Snort -
 - IP deragment: spp_frag2 preprocessor
 - TCP reassembly: spp_stream4 preprocessor

☀ TCP connection state

- Data is come from client or server
 - alert tcp any any -> any 21 (msg: "FTP CWD ~root"; content: "CWD ~root"; sid:336; flow: to_server;)
 - alert tcp any 21 -> any any (msg: "FTP bad login"; content: "530"; flow: from_server;)
- Snort
 - spp_stream4 preprocessor
 - :sp_clientserver detection-plugins

How to IDSs detect intrusions (3/6)

☀ Protocol-decodes (Protocol-analysis)

- Break down a packet into individual fields
 - Alert icmp any any -> any any (msg: "ICMP PING NMAP"; dsize:0; itype:8; sid:469;)
- Snort
 - IP, TCP, UDP, ICMP decodes
 - Detection plugins: sp_icmp_code_check, sp_icmp_id_check, sp_icmp_seq_check, sp_icmp_type_check

☀ Application-layer Preprocessors/normalizers

- Create some sort of "common" form
- Evade rule-1
 - alert tcp any any -> any 21 (msg: "FTP CWD ~root"; content: "CWD ~root"; sid:336; flow: to_server;)
 - CWD ~root or CWD ~root
- Evade rule-2
 - Alert tcp any any -> any 80 (msg: "WEB-CGI phf access"; content: "/phf"; sid: 886)
 - http:// example.com/cgi-bin/%2F%70%68%66?Qalias=%0Acat+/etc/password
- Snort
 - Processors: spp_httpinspect, spp_rpc_decode, spp_telnet_negotiation ...

How to IDSs detect intrusions (4/6)

- ★ State-based application-layer protocol decode
 - Understand the state of application-layer protocol
 - Command mode and data mode switching
 - Alert tcp any any -> any 25 (msg: “SMTP vrfy decode”; content “VRFY decode”; sid: 672)
 - Snort
 - spp_httpinspect, spp_rpc_decode, spp_telnet_negotiation preprocessors

How to IDSs detect intrusions (5/6)

☀ Protocol-field pattern-match

- ☀ Decoding all the fields in the packets restrict searches only to the valid contexts
 - alert tcp any any -> any 80 (msg: "WEB-CGI phf access";
uricontent: "/phf"; sid: 886;)
- ☀ Snort
 - spp_httpinspect, spp_rpc_decode

☀ Protocol-validation, compliance-testing, RFC checking

- ☀ Detection 0-day exploits
- ☀ DNS query, HTTP method, SMTP ...

How to IDSs detect intrusions (6/6)

☀ Other detection method

☀ Protocol anomaly detection

- Too long request, overflow attempt

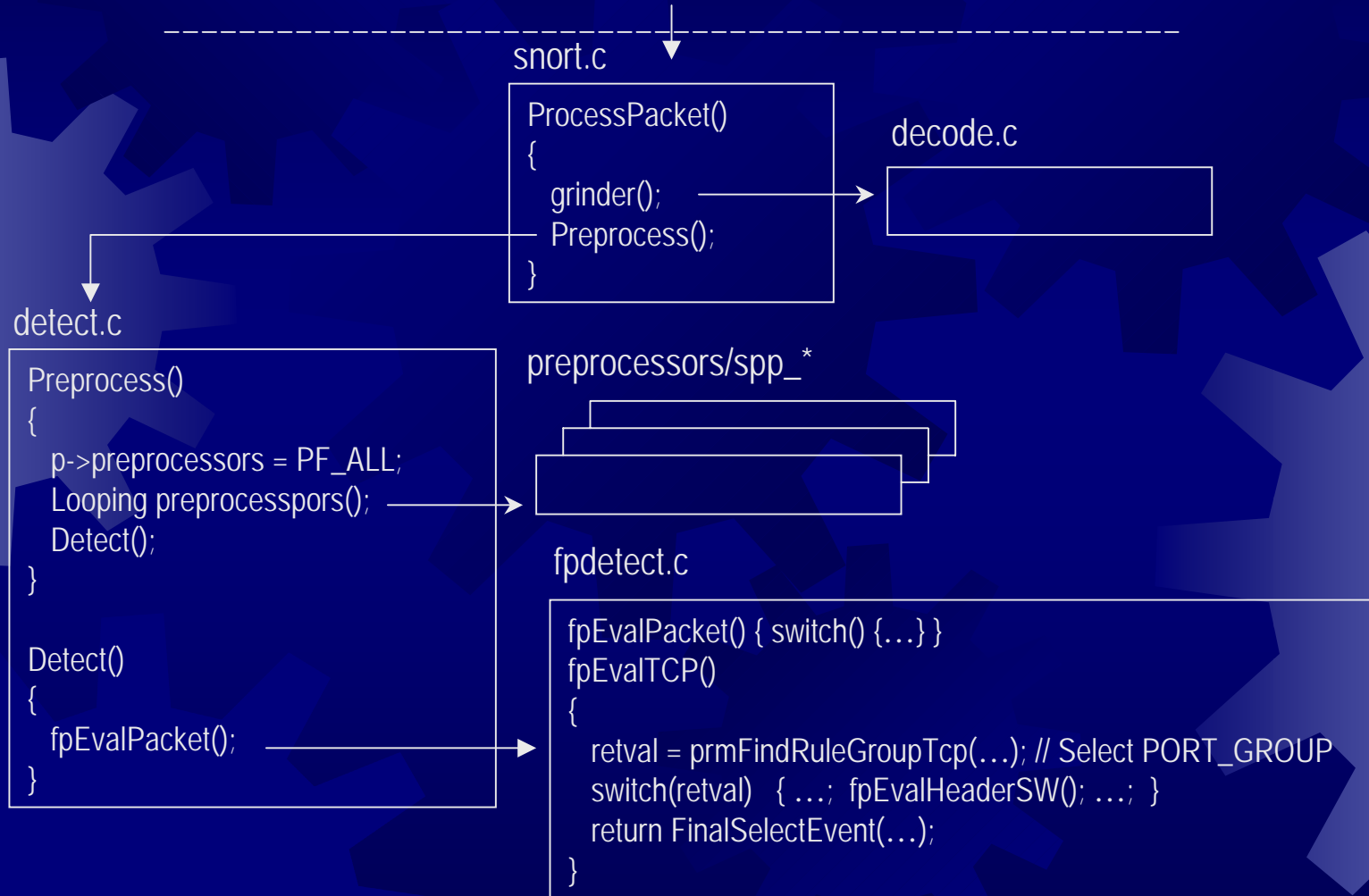
☀ Baseline anomaly detection

- SYN flooding, UDP flooding

☀ Heuristics and behavior analysis

- Login failed times ≥ 3 in short time

Snort's Inner Workings (1/4)



Snort's Inner Workings (2/4)

fpdetect.c

```
fpEvalPacket() { switch() {...} }
fpEvalTCP()
{
    retval = prmFindRuleGroupTcp(...); // Select PORT_GROUP
    switch(retval) { ...; fpEvalHeaderSW(); ...; }
    return FinalSelectEvent(...);
}
fpEvalHeaderSW()
{
    // multip-pattern detection engine
    mpseSearch(, ontx_match, );
    // No-content rules matching
    for(...)
    {
        fpEvalOTN();
        fpEvalRTN();
    }
}
```

ListHead (Alert, Pass, Log, Activation, Dynamic)

```
RuleTreeNode *IpList;  
RuleTreeNode *TcpList;  
RuleTreeNode *UdpList;  
RuleTreeNode *IcmpList;
```

alert tcp any any → any 21 (msg: “FTP CWD ~root”;
content: “CWD ~root”; flow: to_server; sid: 336;)

RuleTreeNode

```
RuleFpList *rule_func;  
Source IP;  
Destination IP;  
Source Port;  
Destination Port;  
RuleTreeNode *right;  
OptTreeNode *down;
```

RuleTreeNode

```
RuleFpList *rule_func;  
Source IP;  
Destination IP;  
Source Port;  
Destination Port;  
RuleTreeNode *right;  
OptTreeNode *down;
```

OptTreeNode

```
OptFpList *opt_func;  
void *ds_list[64];  
OptTreeNode *next;
```

OptFpList

```
void *context;  
int (*OptTestFunc)(...);  
OptFpList *next;
```

OptTreeNode

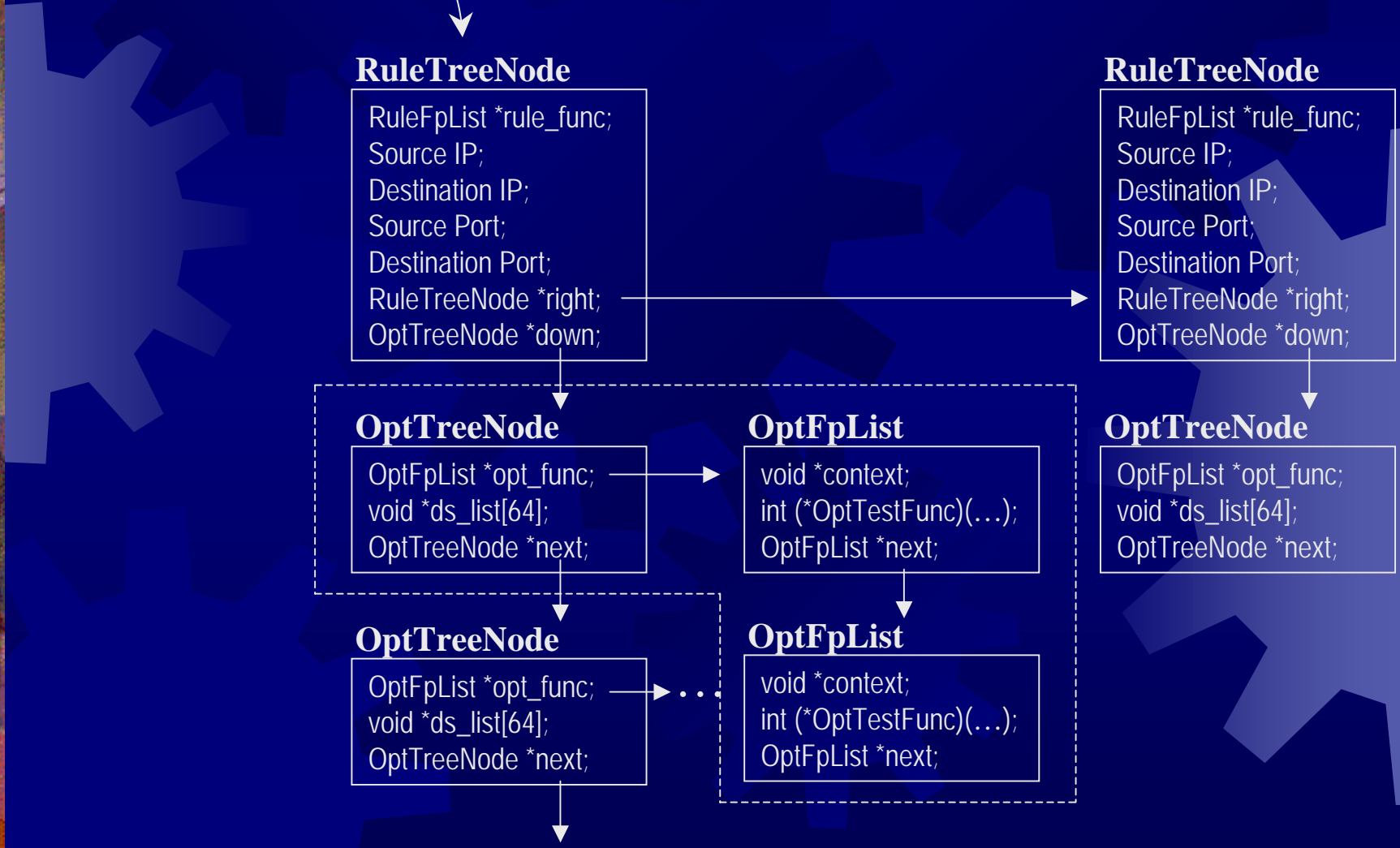
```
OptFpList *opt_func;  
void *ds_list[64];  
OptTreeNode *next;
```

OptTreeNode

```
OptFpList *opt_func;  
void *ds_list[64];  
OptTreeNode *next;
```

OptFpList

```
void *context;  
int (*OptTestFunc)(...);  
OptFpList *next;
```



PORT_RULE_MAP

```
PORT_GROUP *prmSrcPort[MAX_PORTS];  
PORT_GROUP *prmDstPort[MAX_PORTS];  
PORT *prmGeneric;
```

PORT_GROUP

```
RULE_NODE *pgHead;  
RULE_NODE *pgHeadNC;  
RULE_NODE *pgUriHead;  
void *pgPatData;  
Void *pgPatDataUri;
```

RULE_NODE

```
RULE_NODE *next;  
RULE_PTR rnRuleData;  
int iRuleNodeID;
```

RULE_NODE

```
RULE_NODE *next;  
RULE_PTR rnRuleData;  
int iRuleNodeID;
```

Multi-Pattern Matching Structure

OptTreeNode

```
OptFpList *opt_func;  
void *ds_list[64];  
OptTreeNode *next;
```

OptFpList

```
void *context;  
int (*OptTestFunc)(...);  
OptFpList *next;
```

OptFpList

```
void *context;  
int (*OptTestFunc)(...);  
OptFpList *next;
```

Playing by the Rules (1/10)

☀ Snort Rule Format

Rule Header

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 6666:7000 \
```

Rule Body

```
(msg:"EXPLOIT CHAT IRC topic overflow";
```

Meta-Data

```
flow:to_client,established;\
```

Non-Payload Detection Options

```
content:"|EB|K|S2|E4 83 C3 0B|K|88 23 B8|Pw"; \
```

Payload Detection Options

```
react: block, msg; tag: session, 10, seconds;\
```

Post-Detection Rule Options

```
reference:bugtraq,573; reference:cve,1999-0672; \
```

Meta-Data

```
classtype:attempted-user; sid:307; rev:9;)
```

Playing by the Rules (2/10)

☀ Rule Header

☀ Rule Action

- Alert – Alert and log packets
 - Log – Log packets
 - Pass – Ignore packets (note: Snort -o : Pass|Alert|Log)
 - Activate – Alert and turn on another dynamic rule
 - Dynamic – remain idle until activated by an activate rule
- ### ☀ Protocols, IP Addresses, Port Numbers, Direction operator

Playing by the Rules (3/10)

☀ Meta-Data Rule Options

- ☀ Provide related information to describe the rule
- ☀ msg, reference, rev, classtype, priority

☀ Payload Detection Rule Options

- ☀ Look for the data inside the packet payload and can be **inter-related**
- ☀ Options
 - content, uricontent, byte_test, pcre
 - depth, offset, distance, within, byte_jump
 - nocase, rawbytes, isdataat

Playing by the Rules (4/10)

★ Payload Detection Rule Options

★ content

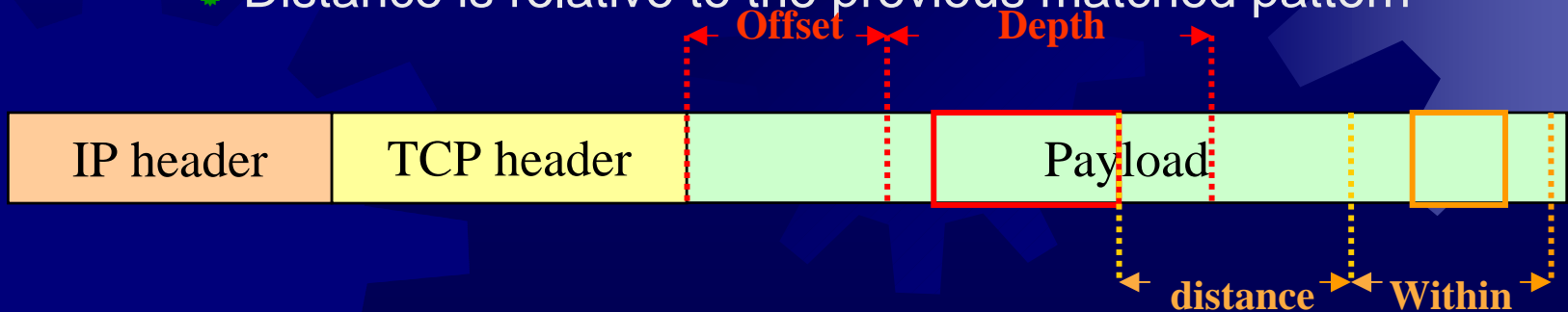
- Search specified content in the payload
- content: “abcde” or content: “|0d0a 0921|”

★ offset, within

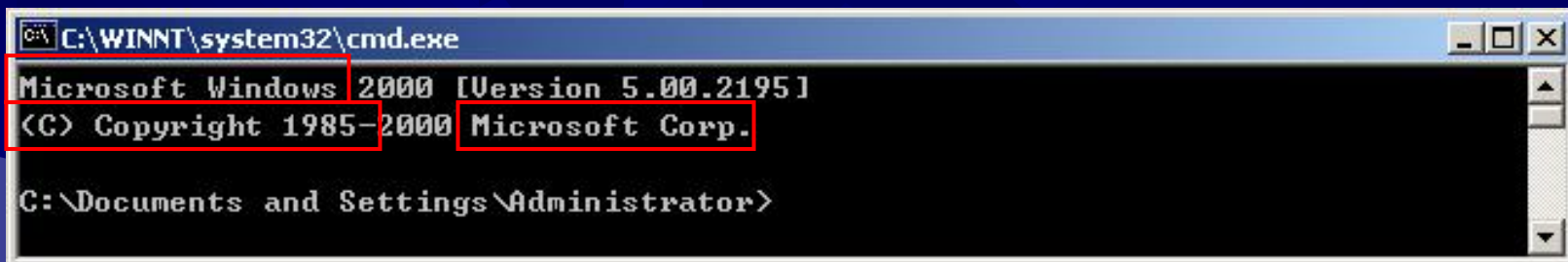
- **Where** to start searching
- Within is relative to the previous matched pattern

★ depth, distance

- **How far** into a packet snort should be search
- Distance is relative to the previous matched pattern

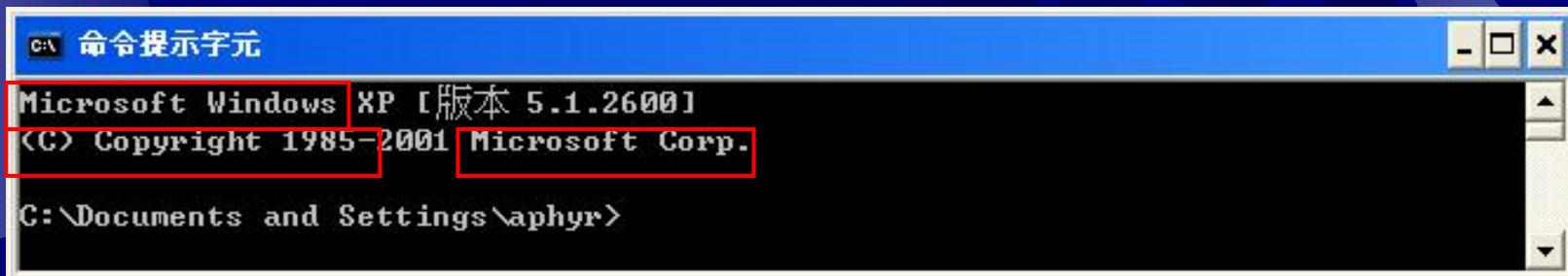


Playing by the Rules (5/10)



```
C:\WINNT\system32\cmd.exe
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\Administrator>
```



```
C:\ 命令提示字元
Microsoft Windows XP [版本 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\aphyr>
```

```
alert tcp $HOME_NET !21:23 -> $EXTERNAL_NET any \
(msg:"ATTACK-RESPONSES Microsoft cmd.exe banner";\
flow:from_server,established;\
content:"Microsoft Windows"; depth:128;\
content:"|28|C|29| Copyright 1985-"; within:60;\
content:"Microsoft Corp."; distance:5; within:15;\
reference:nessus,11633; classtype:successful-admin; sid:2123; rev:2;)
```

Playing by the Rules (6/10)

★ Payload Detection Rule Options

★ byte_test

- Testing binary values or converting representative bytes testing
- byte_test: <bytes_to_convert>, <operator>, <value>, <offset> ,[, [relative], [big], [little], [string], [hex], [dec], [oct]]

★ byte_jump

- Grap bytes, convert them to values and jump up that many bytes
- byte_test: <bytes_to_convert>, <operator>, <offset> ,[, [relative], [big], [little], [string], [hex], [dec], [oct], [align]]

No.	Time	Source	Destination	Protocol	Info
7	0.004390	10.9.65.207	10.9.65.81	SMB	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: 312
8	0.005515	10.9.65.81	10.9.65.207	SMB	Session Setup AndX Request, NTLMSSP_AUTH
9	0.006934	10.9.65.207	10.9.65.81	SMB	Session Setup AndX Response, Error: STATUS_LOGON_FAILURE
10	0.008555	10.9.65.81	10.9.65.207	TCP	3039 > microsoft-ds [FIN, ACK] Seq=3860964124 Ack=3335605350

```

Frame 8 (352 bytes on wire, 352 bytes captured)
Ethernet II, Src: 00:50:ba:75:d7:6f, Dst: 00:03:47:73:d9:33
Internet Protocol, Src Addr: 10.9.65.81 (10.9.65.81), Dst Addr: 10.9.65.207 (10.9.65.207)
Transmission Control Protocol, Src Port: 3039 (3039), Dst Port: microsoft-ds (445), Seq: 3860963826, Ack: 3335605350, Len:
NetBIOS Session Service
SMB (Server Message Block Protocol)
  SMB Header
    Session Setup AndX Request (0x73)
      word Count (wct): 12
      AndXCommand: No further commands (0xff)
      Reserved: 00
      AndXoffset: 294
      Max Buffer: 4356
      Max Mpx Count: 10
      VC Number: 0
      Session Key: 0x00000000
      Security Blob Length: 162
      Reserved: 00000000
    Capabilities: 0xa00000d4
  Byte Count (bcc): 235
  Security Blob: 4E544C4D535350000300000018001800...
    Native OS: windows 2002 2600
    Native LAN Manager: windows 2002 5.1
    Primary Domain:
  
```

```

alert tcp $EXTERNAL_NET any -> $HOME_NET 139\
(msg:"NETBIOS SMB NTLMSSP_AUTH unicode overflow attempt";\
flow:to_server,established;\
content:"|FF|SMB|73|"; depth:5; offset:4; nocase;\
byte_test:1,>,127,6,relative;\
byte_jump:1,32,relative,word;\
byte_test:2,>,512,0,relative;)
  
```

0000	00	03	47	73	d9	33	00	50	ba	75	d7	6f	08	00	45	00	..Gs.3.P	.u.o..E.
0010	01	52	06	bc	40	00	80	06	5b	b8	0a	09	41	51	0a	09	.R..@...	[...AQ..
0020	41	cf	0b	df	01	ba	e6	21	a1	f2	c6	d1	4c	66	50	18	A.....!LFP.
0030	f9	78	ac	f8	00	00	00	00	01	26	ff	53	4d	42	73	00	.x.....	..&.SMBs.
0040	00	00	00	18	07	c8	00	00	00	00	00	00	00	00	00	00&..
0050	00	00	00	00	ff	fe	00	08	20	00	0c	ff	00	26	01	04&..
0060	11	0a	00	00	00	00	00	00	00	a2	00	00	00	00	00	d4
0070	00	00	a0	eb	00	4e	54	4c	4d	53	53	50	00	03	00	00	...NTL	MSSP...
0080	00	18	00	18	00	62	00	00	00	18	00	18	00	7a	00	00b..Z..
0090	00	0c	00	0c	00	40	00	00	00	0a	00	0a	00	4c	00	00@..L..
00a0	00	0c	00	0c	00	56	00	00	00	10	00	10	00	92	00	00v..
00b0	00	15	82	88	e0	43	00	4f	00	4d	00	50	00	41	00	51C.O	.M.P.A.Q
00c0	00	61	00	70	00	68	00	79	00	72	00	43	00	4f	00	4d	.a.p.h.y	.r.C.O.M
00d0	00	50	00	41	00	51	00	0e	1e	bb	d3	4d	45	b2	96	00	.P.A.Q..	...ME..

Playing by the Rules (8/10)

★ Payload Detection Rule Options

★ rawbytes

- Look at the raw packet data, ignoring any decoding by preprocessing

★ uricontent

- Searches the NORMALIZED request URI field

★ pcre (Perl Compatible Regular expression)

- Using regular expression to search payload or URI field
- pcre: [!]/regex/ismxAEGRUB

Playing by the Rules (9/10)

★ Non-Payload Detection Rule Options

- ★ Look for non-payload data
- ★ IP fields: fragoffset, ttl, id, ipopts, fragbits, ip_proto
- ★ TCP fields: flags, seq, ack, window
- ★ ICMP fields: icode, icmp_id, icmp_seq
- ★ Flow: to_client, to_server, established, stateless, no_stream, only_stream
- ★ Flowbits: set, isset

Playing by the Rules (10/10)

★ Post-detection

- Triggers that happen after a rule fired
- Logto – output to file
- Session – extract user data from TCP sessions
- Sesp – close sessions (reset)
- React – close connection and send visible messages (to browser)
- Tag – allow to log more than one packets after a rule is fired

Discussion

