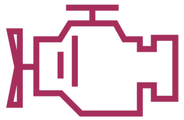
Snort Pre-processors

Joe Abraham
Cybersecurity Consultant

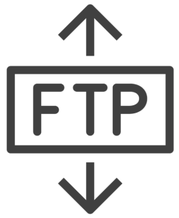
@joeabrah www.defendthenet.com

What's a Pre-processor?



Rule engine matches traffic to rules

Pre-processors are modular plugins that handle packets



FTP/Telnet pre-processor provides stateful inspection for FTP and Telnet

Some Other Pre-processors

Stream

Tracks TCP and UDP sessions

Sensitive Data

Detects and filters Personally Identifiable Information (PII)

SSL/TLS

Decodes and filters SSL and TLS traffic

HTTP Inspect

sfPortscan

AppID

Additional Important Pre-processors

The HTTP Inspect Pre-processor

The HTTP Inspector

HTTP message sections detected:

Request line

Status line

Headers

Content-length message body

Chunked message body

Previous message body

Trailers

Header and URI Inspection

Normalizes header fields

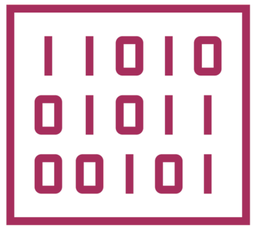
URI type detection:

- **Asterisk**
- **Authority**
- **Origin**
- **Absolute**

Provides dissected URI information

Alert on aspects of the HTTP conversations

HTTP Inspector Rules



(119:19) HTTP header line exceeds 4096 bytes

(119:33) HTTP request URI has space character that is not percent-encoded

(119:239) Expect header sent without a message body

HTTP Inspect Configurations

balanced.lua

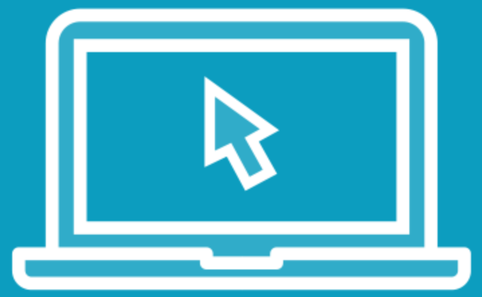
```
------------------------------------------------------------------
-----------
-- balanced connectivity and security policy
-- use with -c snort.lua --tweaks balanced
------------------------------------------------------------------
-----------

arp_spoof = nil

detection = { pcre_override = false }

http_inspect.request_depth = 300
http_inspect.response_depth = 500

port_scan = nil

stream_ip.min_frag_length = 16

table.insert(
    binder, 1, -- add http port binding to appease the perf gods
    { when = { proto = 'tcp', ports = '80', role='server' }, use
= { type = 'http_inspect' } })
```

connectivity.lua

```
------------------------------------------------------------------
-----------
-- reduced security policy that favors connectivity
-- use with -c snort.lua --tweaks connectivity
------------------------------------------------------------------
-----------

arp_spoof = nil

http_inspect.request_depth = 300
http_inspect.response_depth = 500

http_inspect.unzip = false
http_inspect.utf8 = false

port_scan = nil

stream_ip.min_frag_length = 16

table.insert(
    binder, 1, -- add http port binding to appease the perf gods
    { when = { proto = 'tcp', ports = '80', role='server' }, use
= { type = 'http_inspect' } })
```

Use the granular controls to
tune the system to your needs!

Demo



Explore HTTP Inspect pre-processor configurations and validate functionality

The Port Scan Pre-processor

How it Works

Detect various scans

Gathers negative responses to probes

ICMP unreachables

TCP RST

Compares stats and request/response data

```
default_med_port_scan =
{
    protos = 'all',
    scan_types = 'all',

    tcp_window = 90,
    udp_window = 90,
    ip_window = 90,
    icmp_window = 90,

    tcp_ports = tcp_med_ports,
    tcp_decoy = tcp_med_decoy,
    tcp_sweep = tcp_med_sweep,
    tcp_dist = tcp_med_dist,

    udp_ports = udp_med_ports,
    udp_decoy = udp_med_decoy,
    udp_sweep = udp_med_sweep,
    udp_dist = udp_med_dist,

    ip_proto = ip_med_proto,
    ip_decoy = ip_med_decoy,
    ip_sweep = ip_med_sweep,
    ip_dist = ip_med_dist,

    icmp_sweep = icmp_med_sweep,
}
```

- Configures protocol and scan types
- Detection interval for scans
- TCP-specific configurations
- UDP-specific configurations
- IP-specific configurations

Port Scan Types

Portscan: One-to-one scanning multiple ports

Portsweep: One-to-many scanning one port

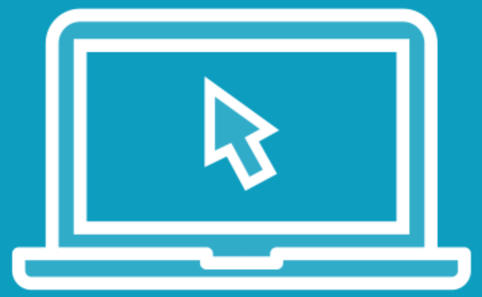
Decoy Portscan: One-to-one with spoofed souce IP addresses

Distributed Portscan: Many-to-one scanning multiple ports

Use the Filters!

For narrow and/or specific use cases, be sure to utilize the filters (scans, rejects, networks, ports)

Demo



Explore Port Scan pre-processor and validate functionality

The Application Pre-processing Capabilities

AppID Pre-processor Features

Provides application identifiers for Snort rule writers to utilize

Outputs statistics for applications and their network usage

Lua language to write custom detectors

AppID Requirements

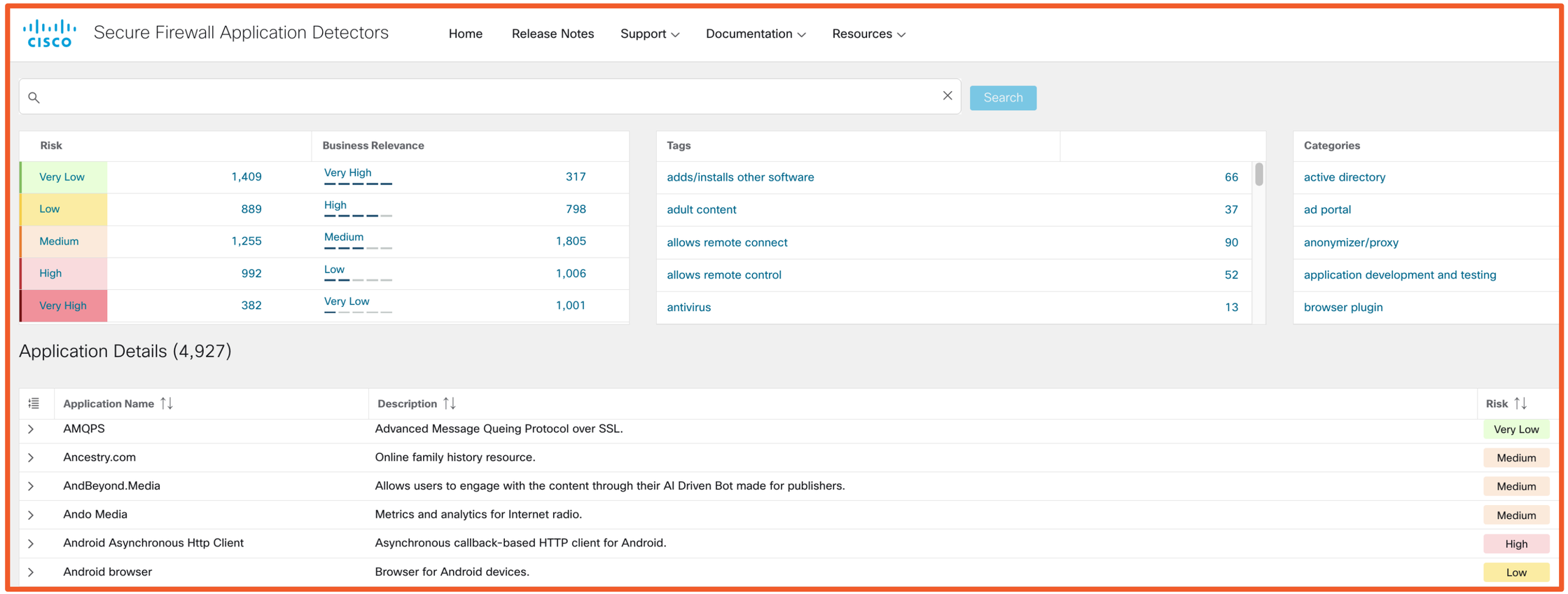
Stream pre-processor is used

Protocol Aware Flushing (PAF) and IP defragmentation is used

HTTP Pre-processor is used

LuaJIT must be installed

Cisco Secure Firewall Application Detectors



https://appid.cisco.com/home

Use this pre-processor to detect and block specific application usage

Application Rule Example

```
alert tcp any any -> any any (msg:”Slack usage detected!";appids:”Slack";sid:10000004;metadata:policy security-ips alert;)
```

Demo

Explore Application pre-processor and validate functionality

HTTP Inspect

sfPortscan

AppID

Module Summary

Additional Documentation

http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node17.html

https://www.cisco.com/c/en/us/td/docs/security/secure-firewall/snort3-inspectors/snort-3-inspector-reference/port-scan-inspector.html

https://www.snort.org/documents

https://appid.cisco.com/home

Up Next:
Snort Plugins