



Real World Security Scripting

Ed Aldridge (GOV) / Chris Sanders (CTR) / Jason Smith (CTR)



What is the SPAWAR NSOC?

- SPAWAR Network Security Operations Center (NSOC)
- Computer Network Defense Service Provider (CNDSP)
- Protect, Detect, Respond, and Sustain
- ~20 Intrusion/Incident Analysts
 - + Cyber Threat Analysis Cell
 - + Fusion Cell
 - + Vulnerability Analysis and Auditing Team
 - + More...

Who Are These Guys?

Chris Sanders

- Packet ninja
- Author of Practical Packet Analysis (No Starch Press)
- CISSP, GCIA, GCIH, GREM

Fusion Team
SPAWAR NSOC

Jason Smith

- RegEx samurai
- Physicist by degree, computer security guy by dumb luck
- GCIA, GCFA, LPIC



Ed Aldridge

- IDS Team Lead
- CISSP, GCIA, GCIH, GCFA, GSEC, RHCE

Objectives

Two Goals:

1. A better understanding of the importance of scripting in analysis and how easy it is to get started with simple tasks
2. Share some really cool and useful scripts that we use every day at SPAWAR

Common Reaction

Boss: It would be nice if we could do something useful with this data....

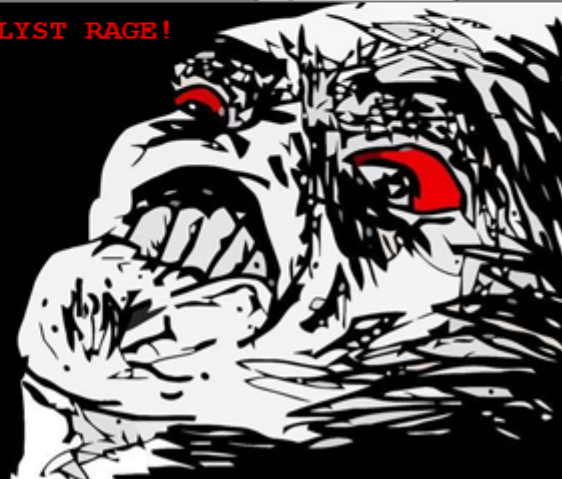


Analyst: I don't like where this is going.



Boss: Maybe you could code something?

ANALYST RAGE!



You may end up in this situation...

CNDSP Operations – Day 1

- Raw PCAP Data
- SiLK Netflow Data

.....

That's it.



Scripting is Important

- Simple scripting can yield big results
- The best way to get the results you need from the data you have
- You won't always have fancy commercial tools
- Help automate tasks or expand capabilities
- A skill that makes you more valuable to your organization

Automation and Capabilities

Automation

- Make repetitive tasks faster
- Typically low effort/high reward
- Easy way to make friends
- Great place to start

Capabilities

- New views on old data
- Varying effort level and reward
- Provide direct value towards mission objectives
- Requires unique ideas

Snort Rule Updater

Problem:

- Takes an analyst 2 minutes per sensor to update custom Snort rules file
- Multiply task times 100 sensors
== 3 ½ hour task



Solution:

- Simple automation task in BASH
- Utilize SSH to update all Snorts sensors from one location

Snort Rule Updater (Workflow)

1. SCP custom rule file to sensor
2. Make backup of existing rule file
3. Replace old file with new file
4. Record changes to log file
5. Restart Snort

Code Disclaimer

- We are NOT programmers!
- Code samples in these slides are brief and incomplete
- Full scripts and code (with comments) are downloadable.
Links to come...



Snort Rule Updater (Variables)

```
sensors="192.168.1.1 192.168.1.2 192.168.1.3 "
```

```
rulepath=/etc/snort/rules
```

```
customrules=custom.rules
```

```
user=snortuser
```

Snort Rule Updater (Error Checking)

```
if [ ! -f $customrules ]; then

    echo "### Error!"

    echo "### In order to use this script a
file named $customrules must exist in the
directory the script was executed from!"

    exit 0

fi
```

Snort Rule Updater (Backup)

```
for ip in $sensors
do
    echo "## Connecting to $ip"
    if ssh -t -q $user@$ip "exit"
    then
        echo "## Creating a backup of current rule
file..."
        ssh -t -q $user@$ip "sudo cp
$rulepath/$customrules $rulepath/$customrules.bak
;exit"
```

Snort Rule Updater (Transfer and Logging)

```
echo -n "## Transferring new rules file to  
sensor..."
```

```
scp -q $customrules $user@$ip:$rulepath
```

```
echo "## Creating log file of rule changes"
```

```
ssh -t -q $user@$ip "sudo diff  
$rulepath/$customrules  
$rulepath/$customrules.bak > \  

```

```
/var/log/snortrules/snort.rule.update.$(date  
+%m%d%y.%H%M%S) ;exit"
```

Snort Rule Updater (Restart Snort)

```
echo "## Restarting Snort"  
  
ssh -t -q $user@$ip "sudo /etc/init.d/snortd  
restart ;exit"  
  
echo "## Rule update completed on $ip"  
  
echo ""  
  
fi  
  
done
```

Snort Rule Updater (Results)

- Before Script – 3 ½ hours
- After Script – 5 Minutes
- End Result – Happy analysts

- Positive Side Effect- Analysts were more likely to update and tweak Snort rules proactively

Problem

- Common “2nd Level” analysis netflow queries are repetitive and time consuming.
- A dozen queries per day.
- ~1 Hour of Processing Time

Solution

- Automate queries on a schedule so they are prepared for analysts at shift start
- Simple Python script



AutoSiLK (Workflow)

1. Run multiple SiLK Queries
2. Write results to file
3. E-mail contents of file

AutoSiLK (Imports and Bash Function)

```
#!/usr/bin/env/python

import sys, os, subprocess, smtplib, time

def runBash(cmd):

    p = subprocess.Popen(cmd, shell=True,
stdout=subprocess.PIPE)

    out = p.stdout.read().strip()

    return out
```

AutoSiLK (Variables)

```
sensors = ["s0", "s1", "s2", "s3", "s4", "s5",  
"s6", "s7"]
```

```
startdate = runBash("date -d '-12 hour'  
+%Y/%m/%d:%H")
```

```
enddate = runBash("date +%Y/%m/%d:%H")
```

```
open('autosilk.data', 'w')
```

```
open('autosilk.data', 'a').write("Top Talkers  
Report for Last 12 Hours.\n\n")
```

AutoSiLK (SiLK Queries)

```
for host in sensors:
```

```
    silkqry1 = runBash("rwfilter --start-  
date=%s --end-date=%s --protocol=1,6,17 --  
sensor=%s --type=all --pass=stdout | rwstats -  
-count=10 --fields=sip,$
```

```
    silkqry2 = runBash("rwfilter --start-  
date=%s --end-date=%s --protocol=1,6,17 --  
sensor=%s --type=all --pass=stdout | rwstats -  
-count=10 --fields=sip,$
```

```
    silkqry3 = runBash("rwfilter --start-  
date=%s --end-date=%s --protocol=1,6,17 --  
sensor=%s --type=all --pass=stdout | rwstats -  
-count=10 --fields=dip,$
```

AutoSiLK (Mailing Results 1)

```
open('autosilk.data','a').write("=====  
=====  
=====\n\n %s - Top Talking IP Pairs by  
Number of Connections\n\n" % (h$
```

```
open('autosilk.data','a').write("%s \n\n" %  
(silkqry1))
```

```
open('autosilk.data','a').write("-----  
-----\n\n %s - Top Utilized  
Source Ports\n\n" % (host))
```

```
open('autosilk.data','a').write("%s \n\n" %  
(silkqry2))
```

```
open('autosilk.data','a').write("-----  
-----\n\n %s - Top Utilized  
Destination Ports\n\n" % (host))
```

```
open('autosilk.data','a').write("%s \n\n" %  
(silkqry3))
```

AutoSiLK (Mailing Results 2)

```
efrom = "toptalkers@nsoc.med.osd.mil"

eto = ["chris.sanders.ctr@nsoc.med.osd.mil" ,
"jason.smith.ctr@nsoc.med.osd.mil"]

esubject = ('Top Talkers %s hour - %s hour' %
(startdate, enddate))

etext = open('autosilk.data', 'r').read()

emessage = ('From: %s\nTo: %s\nSubject:
%s\n%s\n' % (efrom, eto, esubject, etext))

s = smtplib.SMTP('smtpserver.mil')

rCode = s.sendmail(efrom, eto, emessage)

s.quit()
```

AutoSiLK (Output)

Top Talkers 2011/06/05:03 hour - 2011/06/05:15 hour - Mozilla Thunderbird

File Edit View Go Message Tools Help

Get Mail Write Address Book Tag

from [redacted] ☆

subject **Top Talkers 2011/06/05:03 hour - 2011/06/05:15 hour** 11:29 AM

to ['chris.sanders.ctr@nsoc.med.osd.mil' ☆, 'jason.smith.ctr@nsoc.med.osd.mil'] ☆ other actions ▾

Top Talkers Report for Last 12 Hours.

██████ - Top Talking IP Pairs by Number of Connections

sIP scc	dIP dcc pro	Records	%Records	cumul_%
[redacted] us	[redacted] us	17	256575	14.799324
[redacted] us	[redacted] us	17	252291	14.552222
[redacted] us	[redacted] us	17	144188	8.316808
[redacted] us	[redacted] us	17	140816	8.122310
[redacted] us	[redacted] us	17	82819	4.777025
[redacted] us	[redacted] us	17	81547	4.703656
[redacted] us	[redacted] us	17	18260	1.050243
[redacted] us	[redacted] us	17	17923	1.033804
[redacted] us	[redacted] us	17	10947	0.631426
[redacted] us	[redacted] us	17	10765	0.620928

██████ - Top Utilized Source Ports

sIP scc	bytes sPort	Records	%Records	cumul_%
[redacted] us	298 53	34994	2.012633	2.012633
[redacted] us	304 53	21735	1.250060	3.262693
[redacted] us	260 53	18954	1.090114	4.352807
[redacted] us	298 53	18951	1.089942	5.442749
[redacted] us	306 53	17881	1.028402	6.471151
[redacted] us	300 53	16797	0.966057	7.437208
[redacted] us	296 53	15699	0.902907	8.340115
[redacted] us	206 53	14288	0.821755	9.161871
[redacted] us	208 53	14017	0.806169	9.968040
[redacted] us	304 53	13691	0.787420	10.755459

AutoSiLK (Results)

- Before Script – ~1 Hour
- After Script – Instant! Results waiting for you at shift start.
- End Result – Happy Analysts

- Positive Side Effect – More analysts became involved with “second level analysis”. Additionally, servers were taxed less.

Grabber

- Problem: Lots of available PCAP data, but stored in 2 minute increment files across many sensor locations (size limitations).
- Solution: Build a BASH script to filter, collect, and compile PCAP files for analysts.



Grabber (Workflow)

1. Launches client script from analyst workstation
2. Invokes script on sensor containing PCAP data
3. Server script filters through 2 minute increment PCAP files
4. Files are merged together and SCP transferred back to client
5. Client script cleans up temporary files

Grabber (Sensor Selection)

```
echo "Please select sensor in which you'd like to  
download pcap from: "
```

```
echo "SITE1"
```

```
echo "SITE2"
```

```
read SERVERNAME
```

```
if [ "$SERVERNAME" = "SITE1" ]; then
```

```
SERVERIP=$(echo "192.168.0.1")
```

```
fi
```

```
if [ "$SERVERNAME" = "SITE2" ]; then
```

```
SERVERIP=$(echo "192.168.0.2")
```

```
fi
```

Grabber (Client Sending Files to the Sensor)

```
scp /home/scriptagent/grabber-s-scriptagent.sh  
scriptagent@$SERVERIP:/home/scriptagent/
```

```
ssh -t scriptagent@$SERVERIP "sudo mv  
/home/scriptagent/grabber-s-scriptagent.sh  
/data/ ; sudo /data/./grabber-s-scriptagent.sh  
; sudo rm /data/grabber-s-scriptagent.sh ;  
exit ; bash"
```

Grabber (Server Side Time Selection 1)

```
echo -e "Please enter the timespan you're  
interested in:"
```

```
echo -e "past hour--- = 1"
```

```
echo -e "custom----- = 6"
```

```
read timeinput
```

Grabber (Server Side Time Selection 2)

```
if [ $timeinput = 1 ]; then  
  
oldyear=$(date -d "-1 hour" +%y)  
  
oldmonth=$(date -d "-1 hour" +%m)  
  
oldday=$(date -d "-1 hour" +%d)  
  
oldhour=$(date -d "-1 hour" +%H)  
  
oldminute=$(date -d "-1 hour" +%M)  
  
newminute=$(date +%M)  
  
newhour=$(date +%H)  
  
newday=$(date +%d)  
  
newmonth=$(date +%m)  
  
newyear=$(date +%y)  
  
fi
```

Grabber (Custom Time Selection 1)

```
if [ $timeinput = 6 ]; then

echo "Please enter time span (YYMMDDhhmm
YYMMDDhhmm) : "

read endtime starttime

if [ ${#endtime} != 10 ] || [ ${#starttime} !=
10 ]; then

echo "Wrong date format for the start or end
time, rerun."

exit
```

Grabber (Custom Time Selection 2)

```
else
```

```
oldyear=$(echo $starttime | cut -c1,2)
```

```
oldmonth=$(echo $endtime | cut -c3,4)
```

```
oldday=$(echo $endtime | cut -c5,6)
```

```
oldhour=$(echo $endtime | cut -c7,8)
```

```
oldminute=$(echo $endtime | cut -c9,10)
```

```
<code snipped>
```

```
fi
```

```
fi
```

Grabber (Multiple Simultaneous Users 1)

```
time2=$(echo  
$oldyear$oldmonth$oldday$oldhour$oldminute)
```

```
time1=$(echo  
$newyear$newmonth$newday$newhour$newminute)
```

```
echo -n "Enter a filename for your results  
(name-it-something-you-will-recognize.pcap)  
and press [ENTER]: "
```

```
read fname
```

```
echo -n "Enter your search string in tcpdump  
format, i.e. host aaa.bbb.ccc.ddd or tcp port  
80 and then press [ENTER]: "
```

```
read qstr
```

Grabber (Multiple Simultaneous Users 2)

```
echo 1
```

```
uniqident=$(date +%s)
```

```
echo $uniqident
```

```
sudo mkdir /tmp/$uniqident/
```

```
echo 2
```

```
oldtime=$(date -d "$oldyear-$oldmonth-$oldday  
$oldhour:$oldminute:01" +%s)
```

```
newtime=$(date -d "$newyear-$newmonth-$newday  
$newhour:$newminute:01" +%s)
```

```
difference=$(( ( $newtime - $oldtime ) / 60 ) + 2  
)
```

```
thedata=$(date -d "$newyear-$newmonth-$newday  
$newhour:$newminute")
```

Grabber (Listing PCAP to be Searched 1)

```
minmin=00

while [ $minmin -lt $difference ]; do

minute=$(date --date="$thedata - $minmin
minutes" +%M)

ahour=$(date --date="$thedata - $minmin
minutes" +%H)

month=$(date --date="$thedata - $minmin
minutes" +%m)

day=$(date --date="$thedata - $minmin minutes"
+%d)
```

Grabber (Listing PCAP to be Searched 2)

```
rem=$(( 10#$minute % 2 ))  
  
if [ $rem != 0 -a $minute -lt 58 ]  
  
then  
  
minute=$(( 10#$minute + 1 ))  
  
if [ $minute -lt 10 ]; then  
  
minute=$(echo 0$minute)  
  
fi  
  
fi
```

Grabber (Listing PCAP to be Searched 3)

```
if [ $minute = 59 ]  
  
then  
  
minute=$(echo "00")  
  
fi  
  
echo "/data/pcap/NSOC-2011-$month-$day-  
$ahour:$minute:01" >>  
/tmp/$uniqident/file.list.temp  
  
tac /tmp/$uniqident/file.list.temp >  
/tmp/$uniqident/file.list  
  
let minmin=minmin+2  
  
done
```

Grabber (Filter the PCAP)

```
for file in $(  
do  
tstamp=`date +%s`  
sudo /usr/sbin/tcpdump -nnr $file $qstr -w  
/tmp/$uniqident/scriptagent.$tstamp.$iteration  
echo "$iteration"  
iteration=$((iteration+1))  
done  
echo 6
```

Grabber (Merge Results)

```
sudo /usr/sbin/mergecap -w /data/working-  
$fname /tmp/$uniqident/scriptagent.*
```

```
echo working-$fname > /data/temp.v
```

```
sudo chmod 755 /data/temp.v
```

```
echo 7
```

Grabber (Finishing the Transfer)

```
scp scriptagent@$SERVERIP:/data/temp.v  
/home/scriptagent/  
  
pcapfile=$(cat /home/scriptagent/temp.v)  
  
scp scriptagent@$SERVERIP:/data/$pcapfile  
/home/scriptagent/  
  
echo "Your file is located at $pcapfile"  
  
rm /home/scriptagent/temp.v
```

Grabber (Results)

- Prior Method: ~30 minutes
- With Script: < 2 minutes

- This process occurs dozens of times every day!

- Bonus Side Effect: Analysts became more thorough and started always looking at PCAP when available

Cargo Drop

Problem:

- Higher level analysis involving PCAP files often required packet payloads without header data
- Used often for manual stream reassembly, entropy analysis, etc.

Solution:

- Josh Wright's article* details using Scapy for this
- A simple Python script automates the process



*<http://www.packetstan.com/2010/11/packet-payloads-encryption-and-bacon.html>

Cargo Drop (Workflow)

1. Identify PCAP file through input argument
2. Use Scapy to extract packet payloads
3. Use Strings to purge binary data (optional)
4. Save output to a file

Cargo Drop (Imports and BASH Function)

```
#!/usr/bin/env/python

import sys, os, subprocess

from scapy.all import *

def runBash(cmd):

p = subprocess.Popen(cmd, shell=True,
stdout=subprocess.PIPE)

out = p.stdout.read().strip()

return out
```

Cargo Drop (Get Input File and Variables)

```
if len(sys.argv) < 2:  
  
print "Usage: ./cargodrop [input pcap file]  
[output text file]"  
  
sys.exit(1)  
  
# Assign variable names for input and output  
files  
  
infile = sys.argv[1]  
  
outfile = sys.argv[2]
```

Cargo Drop (Extract Payload with Scapy)

```
fp = open("stage1", "wb")

def handler(packet):

fp.write(str(packet.payload.payload.payload))

sniff(offline=infile, prn=handler, filter="tcp port
80")

if os.path.isfile("stage1"):

print "## Stage 1: Payload successfully extracted!"

else:

print "!! Stage 1: Payload extraction failed!"

sys.exit(1)
```

Cargo Drop (Purge Binary Data)

```
print "## Stage 2: Purging binary data..."  
  
runBash("strings stage1 > %s" % (outfile))  
  
if os.path.isfile(outfile):  
  
print "## Stage 2: Binary data successfully  
purged!"  
  
else:  
  
print "!! Stage 2: Binary data purge failed!"  
  
runBash("rm -rf stage1")
```

Cargo Drop (Results)

- Before Script – No Capability
 - After Script – Capability Achieved
 - End Result – Happy Analysts
-
- Positive Side Effect - Several more tools were written that leverage Cargo Drop

Malfind

Problem:

- Analysts needed the capability to compare our traffic against known malicious domains and IP addresses.

Solution:

- A mix of simple scripts that allow for automated intelligence gathering and malicious activity detection.

Malfind (Workflow)

1. Retrieve IP/Domain lists from open source intelligence sites (MalwareDomainList, ZeusTracker, etc)
2. Scan SiLK data for IP matches from lists
3. Scan PCAP data for domain matches from lists
4. Send matches as e-mail results for further investigation

Malfind (Retrieve and Format Intel [Simple])

```
curl  
  http://www.malwaredomainlist.com/hostslist  
  /hosts.txt >  
  /home/scriptagent/malfind/mdlhosts.content
```

```
cat  
  /home/scriptagent/malfind/mdlhostfile.cont  
  ent | sed 1,6d | awk '{print $2}' >  
  mdlhostfile.hosts
```

Malfind (Expanded Intel Gathering 1)

```
testpage=$(curl -s
  http://www.malwaredomainlist.com/hostslist/ip.txt
  | grep "href")

LENTestpage=$(echo ${#testpage})

echo $LENTestpage

if [ $LENTestpage -gt 0 ]; then

  echo "Malfind could not find
  \"http://www.malwaredomainlist.com/hostslist/ip.
  txt\" or it has changed and is not reporting."

else
```

Malfind (Expanded Intel Gathering 1)

```
else

curl
http://www.malwaredomainlist.com/hostslist/ip.txt
t > /home/scriptagent/malfind/mdlhosts.content

sed '/^$/d' mdlhosts.content >
mdlhosts.content.temp

mv mdlhosts.content.temp mdlhosts.content

fi

dos2unix /home/scriptagent/malfind/mdlhosts.content
```

Malfind (Checking SiLK 1)

```
timeold=$(date -d "-6 hour" +20%y/%m/%d:%H)
```

```
timenew=$(date +20%y/%m/%d:%H)
```

```
/usr/local/bin/rwfilter --start-date=$timeold --end-date=$timenew --data-rootdir=/data/flow -not-address=127.0.0.1 --type=all --pass=stdout |  
/usr/local/bin/rwcut -fields=1 | sed s/.$// |tr -  
d ' '|sed 1d |uniq | sort -u >  
/home/scriptagent/malfind/silk.hosts.s
```

```
/usr/local/bin/rwfilter --start-date=$timeold --end-date=$timenew --data-rootdir=/data/flow -not-address=127.0.0.1 --type=all --pass=stdout |  
/usr/local/bin/rwcut -fields=2 | sed s/.$// |tr -  
d ' '|sed 1d |uniq | sort -u >  
/home/scriptagent/malfind/silk.hosts.d
```

Malfind (Checking SiLK 2)

```
mals=$(grep -xFf  
    /home/scriptagent/malfind/mdlhosts.content  
    /home/scriptagent/malfind/silk.hosts.s  
    |sort -u)
```

```
mald=$(grep -xFf  
    /home/scriptagent/malfind/mdlhosts.content  
    /home/scriptagent/malfind/silk.hosts.d  
    |sort -u)
```

```
LENmals=$(echo ${#mals})
```

```
LENmald=$(echo ${#mald})
```

Malfind (Checking SiLK 3)

```
if [ $LENmals != 0 -o $LENmald != 0 ]; then

    echo "#####"
    >>/home/scriptagent/malfind/alertmail.txt

    echo "# malwaredomainlist.com results #"
    >>/home/scriptagent/malfind/alertmail.txt

    echo
    "#####">>/home/scrip
tagent/malfind/alertmail.txt

fi
```

Malfind (Checking SiLK 4)

```
if [ $LENmals != 0 ]; then
    echo "$mals"
    >>/home/scriptagent/malfind/alertmail.txt
    echo " "
    >>/home/scriptagent/malfind/alertmail.txt
fi

if [ $LENmald != 0 ]; then
    echo "$mald"
    >>/home/scriptagent/malfind/alertmail.txt
    echo " "
    >>/home/scriptagent/malfind/alertmail.txt
fi
```

Malfind (Exclusion Lists 1)

```
grep -xFf  
/home/scriptagent/malfind/custom/exclusion  
.list  
/home/scriptagent/malfind/silk.hosts.s  
|sort -u >  
/home/scriptagent/malfind/fexempt.list ;  
cat /home/scriptagent/malfind/fexempt.list  
/home/scriptagent/malfind/silk.hosts.s |  
sort | uniq -u >  
/home/scriptagent/malfind/tempsilk.list ;  
mv /home/scriptagent/malfind/tempsilk.list  
/home/scriptagent/malfind/silk.hosts.s
```

Malfind (Exclusion Lists 2)

```
grep -xFf
/home/scriptagent/malfind/custom/exclusion
.list
/home/scriptagent/malfind/silk.hosts.d
|sort -u >
/home/scriptagent/malfind/fexempt.list ;

cat /home/scriptagent/malfind/fexempt.list
/home/scriptagent/malfind/silk.hosts.d |
sort | uniq -u >
/home/scriptagent/malfind/tempsilk.list ;
mv /home/scriptagent/malfind/tempsilk.list
/home/scriptagent/malfind/silk.hosts.d
```

Malfind (Rundown 1)

```
thedata=$(date)
```

```
minmin=00
```

```
while [ $minmin -lt 360 ]; do
```

```
minute=$(date --date="$thedata - $minmin  
minutes" +%M)
```

```
ahour=$(date --date="$thedata - $minmin  
minutes" +%H)
```

```
month=$(date --date="$thedata - $minmin  
minutes" +%m)
```

```
day=$(date --date="$thedata - $minmin minutes"  
+%d)
```

Malfind (Rundown 2)

```
rem=$(( 10#$minute % 2 ))  
  
if [ $rem != 0 -a $minute -lt 58 ]  
  
then  
  
minute=$(( 10#$minute + 1 ))  
  
if [ $minute -lt 10 ]; then  
  
minute=$(echo 0$minute)  
  
fi  
  
fi
```

Malfind (Rundown 3)

```
let minmin=minmin+2
```

```
done
```

```
dos2unix malstr.$1.raw
```

```
mac2unix malstr.$1.raw
```

```
cat malstr.$1.raw | sed ':a;N;$!ba;s/\n//g' >  
malstr.$1.new
```

```
perl malstr.pl $1
```

```
date
```

Malfind (Rundown 4)

```
if [ $minute = 59 ]  
  
then  
  
minute=$(echo "00")  
  
fi  
  
sudo /usr/sbin/tcpdump -qnns 0 -A -r  
  /data/pcap/NSOC-2011-$month-$day-  
  $ahour\: $minute\:01 'tcp port 80' | grep -  
  B 10 "Host: " >> malstr.$1.raw  
  
let minmin=minmin+2  
  
done
```

Malfind (Malstr.pl 1)

```
open (MYINPUTFILE1, "<malstr.$ARGV[$1].new");  
  
open (MYOUTPUTFILE1, ">$ARGV[$1].malstr");  
  
while (<MYINPUTFILE1>)  
  
{  
  
@line1 = $_ =~ /(\d{2}\:\d{2}\:\d{2}\.\d{6}) IP  
  (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\.\d{1,5} >  
  (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\.\d{1,5} .{5,  
700}[^ ](Host: .{1,60})\-\- /g;  
  
$linecount = ($#line1 + 1);
```

Malfind (Malstr.pl 2)

```
for ($i = 0; $i <= $linecount; $i = $i + 4)
{
for $time (@line1[$i])
{
for $sip (@line1[$i + 1])
{
for $dip (@line1[$i + 2])
{
for $host (@line1[$i + 3])
```

Malfind (Malstr.pl 3)

```
{  
  
print MYOUTPUTFILE1 "$time - host $sip and host $dip  
- $host \n";  
  
}}}}}  
  
}  
  
close (MYINPUTFILE1) ;  
  
close (MYOUTPUTFILE1) ;
```

Malfind (Formatting E-Mail Results 1)

```
echo "Referencing MalwareDomainList.com domains..."

baddie=$(grep -Ff
    /home/scriptagent/malfind/mdlhostfile.hosts
    /home/scriptagent/malfind/$1.malstr |sort -u)

LENbad=$(echo ${#baddie})

if [ $LENbad != 0 ]; then

    echo
    "#####"
    "#####"
    >>/home/scriptagent/malfind/alertmail.txt

    echo "# malwaredomainlist domain results -
investigate immediately # $2-($1)"
    >>/home/scriptagent/malfind/alertmail.txt
```

Malfind (Formatting E-Mail Results 2)

echo

```
"#####  
#####"
```

```
>>/home/scriptagent/malfind/alertmail.txt
```

```
echo "$baddie - Investigate Immediately"
```

```
>>/home/scriptagent/malfind/alertmail.txt
```

```
#mail -r Malfind -s "$time Megabad alert! -  
$baddie" jasonasmith.ia@gmail.com <  
/home/scriptagent/malfind/alertmail.txt
```

```
echo " "
```

```
>>/home/scriptagent/malfind/alertmail.txt
```

fi

Malfind (Handling Multiple Sensors 1)

```
date
```

```
time=$(date +%d/%m/%y-%H:%M)
```

```
echo "*** The following results are potential  
blacklist matches in our SiLK records ***"  
>/home/scriptagent/malfind/alertmail.txt
```

```
echo " " >>/home/scriptagent/malfind/alertmail.txt  
  
/home/scriptagent/malfind/./malfind-silk
```

```
echo " " >>/home/scriptagent/malfind/alertmail.txt
```

```
echo
```

```
"
```

```
"
```

```
>>/home/scriptagent/malfind/alertmail.txt
```

Malfind (Handling Multiple Sensors 2)

```
echo "*** The following results are potential blacklist
domain matches in our PCAP data ***"
>>/home/scriptagent/malfind/alertmail.txt

/home/scriptagent/malfind/./malfind-c-mailer
192.168.0.1 SITE1&

/home/scriptagent/malfind/./malfind-c-mailer
192.168.0.2 SITE2&

wait

mail -s "Malfind Alerts $time"
jason.smith.ctr@nsoc.med.osd.mil,chris.sanders.ctr@
nsoc.med.osd.mil -- -f Malfind <
/home/scriptagent/malfind/alertmail.txt

rm /home/scriptagent/malfind/custom/threat.dot.list

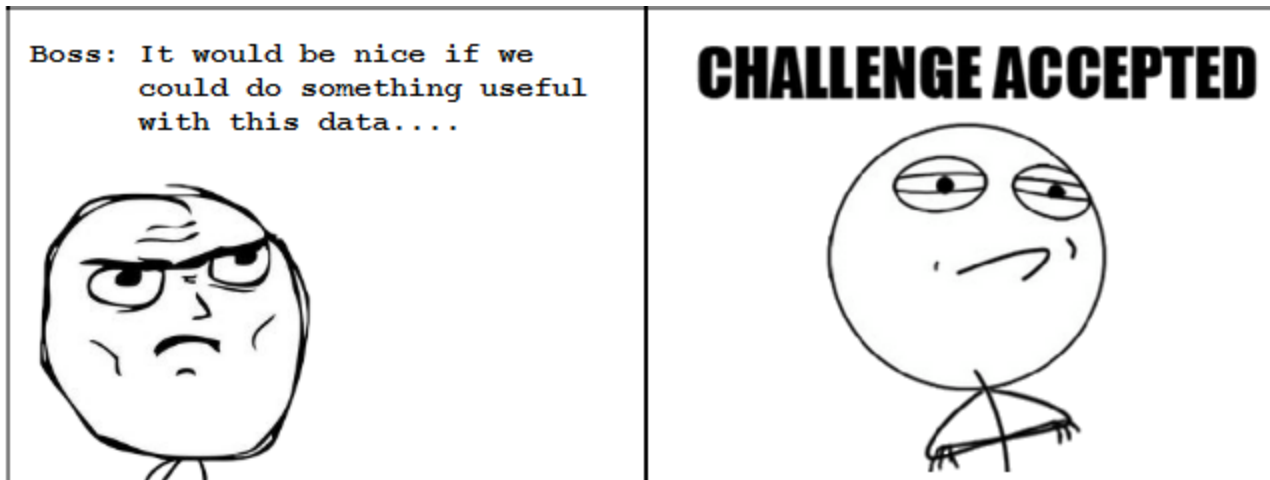
rm /home/scriptagent/malfind/custom/threat.space.list
```

Malfind (Results)

- Before Script – No Open Source Intel Based Detection Capability
- After Script – Capability Achieved
- End Result – A significant number of new CAT7 incidents were found

- Positive Side Effect – Site admins were pleased our analysts were outperforming their desktop antivirus

Final Organizational Result



Source Code

All source code available for download at:

<https://www.forge.mil/>

Search for SPAWAR NSOC

Documentation is there too!

We Need Your Help!

- These tools are crude and inefficient, but they work
- More ideas than time
- We need people to chip in and make them more efficient and increase their capability



Thanks for Staying Awake!

Ed Aldridge

ed.aldridge@nsoc.med.osd.mil

Chris Sanders

chris.sanders.ctr@nsoc.med.osd.mil

Jason Smith

jason.smith.ctr@nsoc.med.osd.mil