SANS

Using Old Tools to Catch Current Adversaries

Mark Jeanmougin, SANS Instructor

Abstract: The presenter will walk through a scenario based on an actual incident he worked. He'll present information about some basic Linux command line tools. You'll get some simple examples of how to chain those tools together into pipelines to solve real world problems. You'll then have the time to explore a dataset to look for adversary activity. Sample questions are provided to guide your trip through the data. Generally, all the information you need is provided in the dataset.

Author: Mark Jeanmougin / markjx@gmail.com / @markjx01
Supplemental material at https://github.com/markjx/oldcurrent

Disclaimer

The information presented here is not intended for use by anyone. If you try to follow along at home and get a hangnail, instantiate global thermonuclear war, or have other adverse side effects: You're on your own. Mark, as well as his past, current, or future employers, family members, and pets disclaim any and all responsibility from now until the end of the Universe.

The author, presenter, and other people who's information is contained in this document disclaim any and all responsibility for any use of any information contained herein.

Lab Requirements

Download data file from:

https://drive.google.com/file/d/15rupTZ6sRDiQOHJCK54O1JTU56Hk4keg/view

You'll need something to uncompress an encrypted 7z file.

- Password: Brunos2Pizza & Test data file: https://github.com/markjx/oldcurrent/raw/main/test.7z
- If you can work with test.7z, then you can work with this lab

You'll need approximately 13GB of disk space for the uncompressed data set

Linux environment (WSL, Virtual Machine, whatever)

For the most recent copy of information about this presentation, please reference:
https://github.com/markjx/oldcurrent/

Link to download access.log.7z:
https://drive.google.com/file/d/15rupTZ6sRDiQOHJCK54O1JTU56Hk4keg/view

Setup for Lab. Please complete these before the start of the Presentation.

1. You'll need a Linux environment for this lab. I tested with Fedora and Ubuntu, but anything should work. It should work under WSL (Windows Subsystem for Linux), a VM (Virtual Machine under Vmware, VirtualBox, KVM/QEMU, Proxmox, etc), or on hardware.
2. You may need root access to install tools. You'll need a tool that can handle password-protected 7z files. To install:
    1. On Fedora: `sudo dnf install -y p7zip`
    2. On Ubuntu: `sudo apt install -y p7zip` or `sudo apt install -y p7zip-full`
    3. If you want to try on high difficulty, feel free to use something like John the Ripper or hashcat instead. ☺ This is **not** recommended.
3. Download the data file from

https://drive.google.com/file/d/15rupTZ6sRDiQOHJCK54O1JTU56Hk4keg/view You'll need approximately 13GB of disk space to uncompress it. You'll get the password for this, at the same time as everyone else, during the presentation ☺

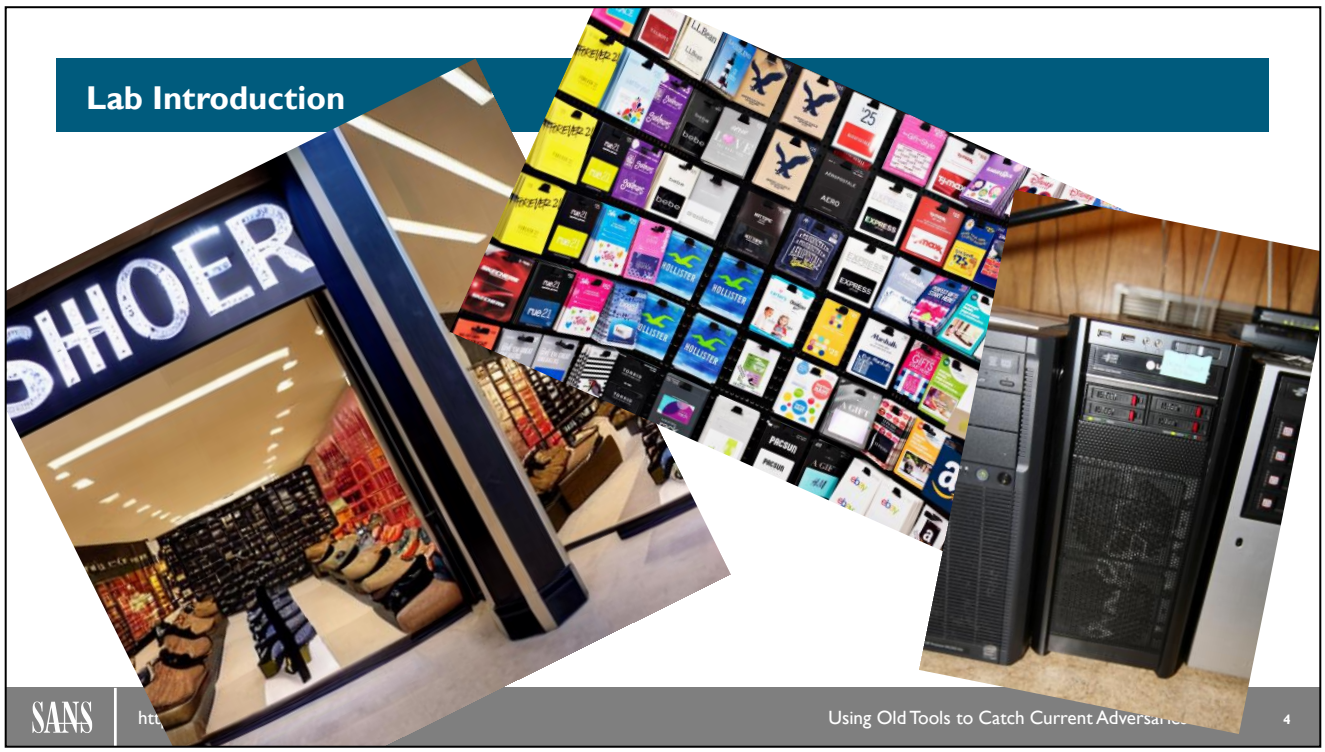
4. If you want to make sure that you're all setup to work with the real data file, I created https://github.com/markjx/oldcurrent/raw/main/test.7z The password is above (the password isn't "above". The password is on the slide, which is above this sentence ☺). If you're able to uncompress this, then you're ready for the real one.
    1. Command to uncompress the test.7z file:
       `7z x test.7z`
       Then enter the password when prompted.
5. You should now be ready for the lab!

All testing was done on x86 / x86_64 systems. But, this should all work from an Apple Mac with an M1/M2/M3 CPU. It would probably work under IRIX with a MIPS CPU. And, it's probably work on a Sun SparcStation under Solaris. AIX on Power and HP-UX would probably work fine too. There's no way it'd work under OS/2, but it'd be fun to write the analysis in REXX…

If you want to attempt on a MacOS system, you may need an additional tool to uncompress the password-protected 7z file.

- Some have had good luck with Keka at https://www.keka.io/en/
- Others used `brew install p7zip` from homebrew at https://brew.sh/

NOTE: Your antivirus scanner *may* flag the test.7z file as a virus. (Specifically Windows Defender may flag it as Trojan:Script/Wacatac.B!ml) It is not a virus. Viruses have to be executable code. This is a password-protected 7zip file. No big deal. There's no executable code.



An adversary was committing gift card fraud. Find it. ☺

Shoe store picture from: https://deepai.org/machine-learning-model/text2img

Gift card picture from: https://www.wired.com/wp-content/uploads/2015/12/gift-cards-holidays-477454060.jpg

Server picture by Mark W. Jeanmougin. All Rights Reserved.

Lab Details

- An adversary was committing gift card fraud at a retailer.
- Suspicion is that they were hitting the www.mygiftcard.com website to find valid gift cards
- Then, selling those gift cards on gift card swap websites
- Unsuspecting buyers were purchasing the gift cards, not knowing they were fraudulent

Agenda

- ✓ Intro
- ❑ whoami
- ❑ Using Old Tools to Catch Current Adversaries
- ❑ Intro to Lab
- ❑ Old Tools
- ❑ Password
- ❑ References
- ❑ Preso to Winning Team

Ask Questions!

When you're watching TV with someone and they rewind to see a funny part, or the make sure they caught a key part of the plot, or rewatch a particularly sporty part of a sports thing, they're also saying this is important to me and I want to share it with you.

$ whoami

- Mark Jeanmougin (markjx@gmail.com / @markjx01)
  - SANS Certified Instructor
  - Security Architect
- Always Blue Team (SOC)
- Digital Forensics & Incident Response
  - Inappropriate Internet Use & Academic Fraud
- IT for >20 years. Security since 2000.

Mark Jeanmougin
markjx@gmail.com / https://twitter.com/markjx01
https://markjx.blogspot.com/
https://github.com/markjx
https://www.linkedin.com/in/markjx

Blue Team for my whole career.
SANS Community Instructor
Digital Forensics & Incident Response
Security Operations Center Analyst & Manager
Started "Experimenting" with UNIX in college.
Been doing IT stuff for over 20 years now. Security since 2000.

While I do have a $DayJob, this work is not endorsed or sponsored by them. But, feel free to reach out to me at mark.jeanmougin@siemens.com

Using Old Tools to Catch Current Adversaries

Give you the tools you'll need to investigate a (simulated) incident

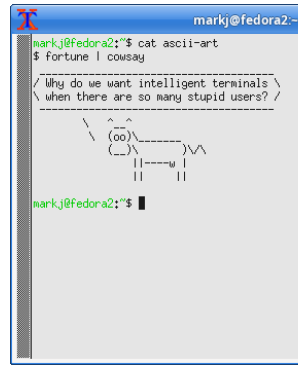
- Linux command line tools
- Web Server Log Analysis

SANS https://github.com/markjx/oldcurrent/

Old Tools

Tools that will come in handy:

- less
- cat
- grep
- sort
- uniq
- head
- tail



Examples:

less access.log

Opens the log file for viewing. PgUp / PgDn and arrow keys should work as expected. g/G goto start/end of a file. / is to search. ? Is reverse search

cat access.log

Displays a file to the screen. Or, reads the file off disk and dumps it into the pipe for processing

grep adversary access.log

Searches through access.log looking for “adversary” and prints any line containing that term. –i is for a case **in**sensitive search.

sort access.log

Sorts the file. –n is to sort by number.

uniq

If a file has duplicate lines, only display them once.

uniq –c

When display lines, prepend the line with number of times that line is seen.

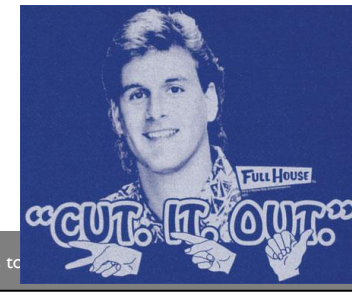
head –n 12 access.log
Prints the first 12 lines

tail –n 12 access.log
Prints the last 12 lines

What the cut?

Pull out the User-Agent from the last 4 lines of an access.log

```
$ tail -n 4 access_log | cut -d \" -f 6
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36
```



cut –d\" –f 6
-d \" When counting fields, use a " as the delimiter

-f 6 Pull the 6th field.

cut –d " " –f 6
-d " " When counting fields, use a space as the delimiter

-f 6 Pull the 6th field.

cut –d, –f 3-7
-d, When counting fields, use a comma as the delimiter

-f 3-7 Pull field 3, 4, 5, 6, & 7

cut –c 12-

-c 12- Starting at character 12, display from character 12 through the end of the line

Tying it all Together

Pull a list of unique User-Agents

```
$ cat access_log | cut -f 6 -d \" | sort | uniq
DavClnt
Microsoft-WebDAV-MiniRedir/10.0.19045
Mozilla/4.0 (compatible; ms-office; MSOffice 16)
Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0
Safari/537.36
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/122.0.0.0 Safari/537.36
```


These slides all build on each other. The yellow highlight indicates what is "new" for this slide.

Explanation:

cat to read the file and send it into the pipe

cut: use the double-quote as the delimiter, pull the 6th field from each line. (In order to figure out your proper delimiter and field number, you'll need to look at the file (maybe using less, head, or tail) and doing some counting on your fingers (and/or toes! ☺)

sort: sorts the lines of the file alphabetically. We're really doing this as a pre-processor step so that uniq works properly. uniq only compares adjacent lines. So, you'll almost always run sort before uniq.

uniq: delete duplicate lines.

A bit more Advanced…

Most Popular User-Agents

```
$ cat access_log | cut -f 6 -d \" | sort | uniq -c
      3 DavClnt
    108 Microsoft-WebDAV-MiniRedir/10.0.19045
      1 Mozilla/4.0 (compatible; ms-office; MSOffice 16)
     65 Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0
Safari/537.36
      6 Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36
```


These slides all build on each other. The yellow highlight indicates what is "new" for this slide.

This slide builds on everything in the previous slide. The new part is adding the "-c" to uniq.

Explanation:
uniq –c: Rather than simply deleting duplicate lines, this shows the **C**ount of each line.

Even more Advanced!

Most Popular User-Agents, sorted

```
$ cat access_log | cut -f 6 -d \" | sort | uniq -c | sort
-n
      1 Mozilla/4.0 (compatible; ms-office; MSOffice 16)
      3 DavClnt
      6 Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36
     65 Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0
Safari/537.36
    108 Microsoft-WebDAV-MiniRedir/10.0.19045
```

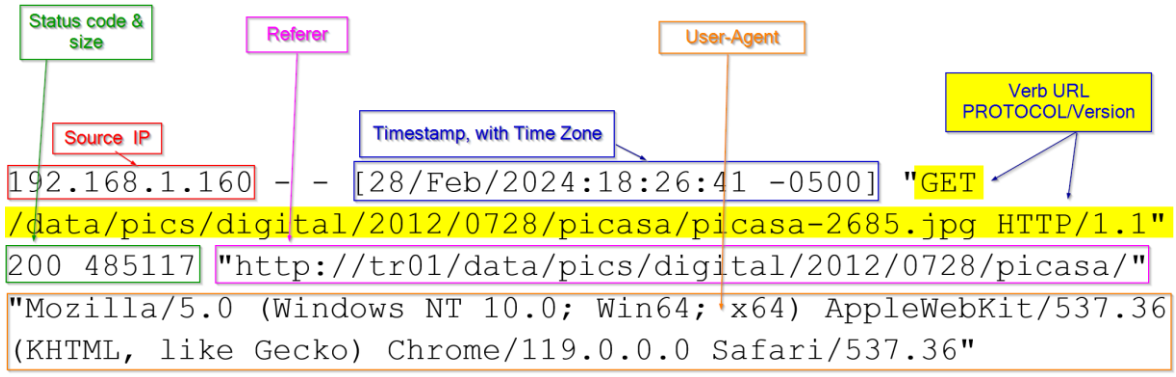

These slides all build on each other. The yellow highlight indicates what is "new" for this slide.

This slide builds on everything in the previous slide. The new part is running the output of the last slide through "sort –n". Now, we obviously (I hope! ☺) see which User-Agent is most popular.

Explanation:

sort –n: **N**umerically sort the lines of text.

Apache / nginx access.log



Example Log Event:

```
192.168.1.160 - - [28/Feb/2024:18:26:41 -0500] "GET
/data/pics/digital/2012/0728/picasa/picasa-2685.jpg HTTP/1.1" 200
485117 "http://tr01/data/pics/digital/2012/0728/picasa/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
```

Fields:

192.168.1.160: Source IP

[28/Feb/2024:18:26:41 -0500]: Time stamp. Day/month/year:hour:minute:seconds Time Zone (-0500 is 5 hours behind UTC)

"GET /data/pics/digital/2012/0728/picasa/picasa-2685.jpg HTTP/1.1": GET (download a file) next is the full path and name of the file; finally HTTP/1.1 says that this is using the HTTP protocol, version 1.1

200: Status code. See https://http.cat/ or https://http.dog/ for more info

485117: Size of transfer

"http://tr01/data/pics/digital/2012/0728/picasa/": Referer. (Yes, this field is called Referer, based on the HTTP header name. Yes,

it is misspelled)

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36": User-Agent

Lab Questions

1. Does the adversary have any legit card number/PIN pairs? How many?
2. When did the malicious activity start?
3. When did the adversary first interact with our site?
4. What else can you tell me about the case?

Lab Questions 2

1. Was the adversary using their home computer to attack us?
2. Was the adversary using a botnet or paying for their attack infrastructure?
3. Could we block some ASN's to drop this traffic? Which ones?
4. Is it likely that would disrupt our paid customers?

Lab Questions 3

1. What's the adversary's IP address?
2. What tribe/group/team does the adversary belong to?
3. How old is the adversary?
4. Is the adversary a "professional" or "script kiddy"?
5. What else can you tell me about the adversary?

Password

Uncompress Command

```
7z x access.log.7z
```

Not here! ☺

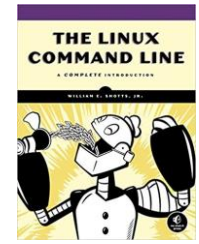
Mark will hand out the compressed file's password verbally in the exercise so everyone gets it at the same time.

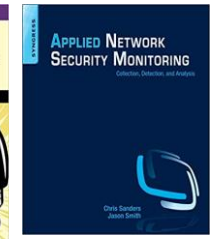
Questions?

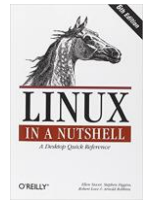
markjx@gmail.com

@markjx01

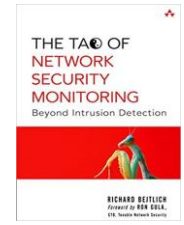
Slide Deck & Scripts:
https://github.com/markjx/oldcurrent

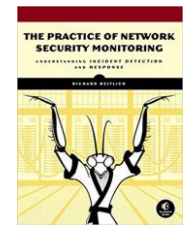






Humble Bundle





Books that may be useful:

- Linux Command Line, 2nd Edition: https://nostarch.com/tlcl2 If you buy from No Starch Press directly, it includes the DRM-free ebook. Support William Shotts, the author, at http://linuxcommand.org/tlcl.php
    - Also may want his *Adventures* book, too.
- Linux in a Nutshell from O'Reilly https://www.amazon.com/Linux-Nutshell-Desktop-Quick-Reference/dp/0596154488/

The Tao of Network Security Monitoring https://www.amazon.com/Tao-Network-Security-Monitoring-Intrusion/dp/0321246772

The Practice of Network Security Monitoring: https://nostarch.com/nsm If you buy from No Starch Press directly, it includes the DRM-free ebook.

Applied Network Security Monitoring by Chris Sanders & Jason Smith. https://www.amazon.com/Applied-Network-Security-Monitoring-Collection/dp/0124172083/

Keep an eye on Humble Bundle (https://www.humblebundle.com/). They periodically do bundles from O'Reilly, No Starch Press, and other great publishers.

Appendix

Things I found interesting, but didn't really have time to include in the presentation

squishycat

cat compressed files

- **gzip**
- **bzip2**
- **lz4**
- **xz**

Or uncompressed!

Photo Credit: My cat, Ceili, just before and after being shaved. Taken by Mark Jeanmougin.
https://github.com/markjx/search2018/

squishycat is like the normal UNIX cat command except: When dealing with normal ASCII text, it just cats it. When dealing with data compressed, it decompresses it first, then cat's it. It currently supports gzip, bzip2, lz4, and xz.

squishycat: Use

```
[markj@tr01 20180415]$ ls S* | while read fn ; do file $fn ; squishycat $fn |
 sha1sum ; echo . ; done
SG_main__470802230000.log: Non-ISO extended-ASCII text, with very long lines
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
SG_main__470802230000.log.bzip2: bzip2 compressed data, block size = 900k
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
SG_main__470802230000.log.gzip: gzip compressed data
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
SG_main__470802230000.log.lz4: LZ4 compressed data (v1.4+)
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
SG_main__470802230000.log.xz: XZ compressed data
6ddafe913bf160a0a6c2c4a949de5270e19682e9  -
.
[markj@tr01 20180415]$
```


Generated compressed files:

```
ifn=SG_main__470802230000.log ;
for i in gzip bzip2 xz lz4
 do
  ofn=${i}.out ;
  (time cat $ifn | $i > ${ifn}.$i) >$ofn 2>&1 &
 done
```