**Computer and Information Security**

**(ECE590-03, Fall 2019, Duke Univ., Prof. Tyler Bletsch)**

**Homework 3**

Name: \_\_\_Nan Ni\_\_\_\_\_\_\_

Duke NetID: \_\_\_\_nn75\_\_\_\_\_\_\_

**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](https://www.alchemistowl.org/pocorgtfo/pocorgtfo03.pdf) (“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: Chapter 3 - User Authentication (8 points)**

(Based in part on review questions from the textbook)

1. In general terms, what are the three means of authenticating a user’s identity?
2. List and briefly describe five possible threats to the secrecy of passwords.
3. What are three common techniques used to protect stored passwords?
4. Which is more important: password complexity requirements or password expiration requirements? Why?
5. What is MFA? What threat model does it deal with?
6. List and briefly describe the principal characteristics used for biometric identification.
7. Describe the general concept of a challenge-response protocol.
8. In a challenge-response password protocol, why is a random nonce used?

**Question 2: File permissions (7 points)**

From Problem 4.5: UNIX treats file directories in the same fashion as files; that is, both are defined by the same type of data structure, called an inode.  As with files, directories include a nine-bit protection string. If care is not taken, this can create access control problems. For example, consider a file with the protection mode 644 (octal) contained in a directory with protection mode 773.

**Explain the interpretation of the 9-bit protection string, for both files and directories. (1)**

**Explain the meaning of the octal protection string of 644 for a file, and 750 for a directory. (2)**

**Do an experiment: Make a directory with permissions 773. Can non-owner non-group users list the contents of the directory? Can such users create new files in the directory? (1)**

Consider the ls output below.

  -rw-r--r--  1 root root 1.6M Sep 25 16:58 gosh.tar.gz

**Explain in detail what each element this line means. (1)**

**What command would you use to change the access rights to allow any user to edit the file? (1)**

**What command would you use to take full ownership from root to yourself?  (1)**

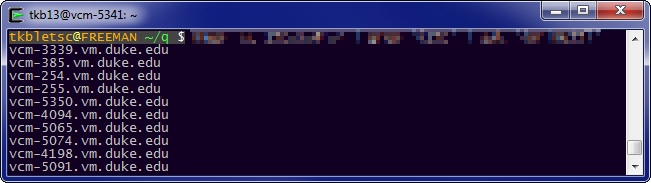
**Question 3: Data processing (16 points)**

In this question, you’ll be manipulating some text data. Beyond the tools covered in class, you may want to look into:

* [AWK](https://9p.io/cm/cs/awkbook/index.html), a general purpose computer language that is designed for processing text-based data, either in files or data streams. The name AWK is derived from the surnames of its authors Alfred V. Aho, Peter J. Weinberger, and Brian W. Kernighan.
* [Perl](https://www.perl.org/) is also a general purpose computer language focused on text-based data developed by Larry Wall.

**Task 1: Filter nmap output (4 points)**

**Using any combination of the tools above, on your Linux VM, write a *single command* with pipes that will parse out only the registered hostnames from an Nmap List scan of the 152.3.64.\* subnet.  Your output should not include those hosts without a hostname or any other extraneous output. Give the full command and a small sample (5 hosts) of output.  Do not use -sT or -sS in the Nmap Scan. A sample screenshot (with command blurred out of course) is shown below:**

****

An attacker might use output like the above to better understand a target environment, especially after they’ve gained a foothold to an internal network.

**Task 2: Log analysis (4 points)**

In Homework 1’s question 2, we reviewed the output of Logwatch. That tool analyzes system logs such as the authorization log **auth.log**. Let’s do a bit of similar analysis ourselves.

**Using wget, obtain** [**this auth.log**](http://people.duke.edu/~tkb13/courses/ece590-sec/homework/auth.log) **from a real production server on the internet.**

**Write a single command that identifies all the *unique* IP addresses that tried and failed to login using the invalid user ‘admin’. Post a screenshot.**

**Task 3: Web app needle in a haystack (8 points)**

A mock web application has been set up here:

<http://target.colab.duke.edu/app/>

The term “web application” here is a misnomer: for safety reasons, there’s no actual server-side code. Rather, there’s “lorem ipsum” content with hyperlinks that branch out to much of the same, with a few hidden things mixed in.

**Using shell-based tools, identify the two pages that have an HTML <form> tag and give the sequence of clicks needed to find these pages from the start. Show your work.**

|  |
| --- |
| **OPTIONAL: For up to 4 points of extra credit, find:**   1. A fat dog 2. A unicorn   As with the main question, give the page with the content and the click sequence needed to get there from the start. Show your work. |

**Question 4: Regular expressions (9 points)**

**NetID validator (3 points)**

Develop a regular expression that matches if and only if the input is a valid NetID (1-3 letters, 1 or more digits).

**URL parser (3 points)**

Develop a regular expression that, for a given URL, matches if and only if it is a Wikipedia page, and it captures just the article part of the URL. Examples:

|  |  |  |
| --- | --- | --- |
| **URL** | **Matches?** | **Match group 1** |
| https://en.wikipedia.org/wiki/Computer\_security | yes | Computer\_security |
| https://duke.edu/ | no | - |

**Pi digits (3 points)**

Develop a shell command to print the MD5 hash of pi (π) up to the first appearance of decimal digits “12345”. To help check your work, the last byte of the answer is 0x02. You can [download the value of pi](https://www.angio.net/pi/digits.html), no need to compute it. For actually employing the regex, you can use grep with -E and other options, perl, Python, or other tools.

**Question 5: GNU Screen (2 points)**

[Screen](http://www.gnu.org/software/screen/) is a full-screen window manager that multiplexes a physical terminal (or GUI terminal window) between several processes, typically interactive shells. When screen is called, it creates a single terminal with a shell in it (or the specified command) and then gets out of your way so that you can use the program as you normally would. Then, at any time, you can create new (full-screen) terminals with other programs in them (including more shells), kill the current window, view a list of the active windows, turn output logging on and off, view the scrollback history, switch between terminals, etc.  All terminals run their programs completely independent of each other. Especially useful for us is the fact you can *detach* a running screen session from your console and let all the associated terminals keep running in the background, then later *reattach* it to another console. Programs continue to run even when screen is detached.

[Review this very brief video intro to screen](https://www.youtube.com/watch?v=hB6Y72DK8mc). Here is a [screen quick reference](http://aperiodic.net/screen/quick_reference).

This might be useful for the MD5 cracking question below, and it’s essential for the problem “John the Ripper” after that.  Start a new screen session on your Kali VM. Prove to yourself that you can detach, reattach, and are generally comfortable with screen.

*Note: If you prefer another terminal virtualizer, such as tmux, you may use that instead.*

**Paste a screenshot below of screen --list two separate screen sessions running.**

**Question 6: Cracking MD5 hashes with hashcat (6 points)**

There is an FTP server called Serv-U.  It is actually a commercial program written by CATsoft.  Hackers sometimes install pirated copies on computers they compromise.  This allows an attacker to access the file system and run commands on the compromised computer.  The FTP server usernames and passwords (used by the attackers) are stored in a configuration file.  The passwords are hashed with MD5 hashing algorithm. Here is a sample file from a compromised machine:

|  |
| --- |
| [USER=admin|1]  **Password=ly5cf2ff593419bcf3d22c62e65a82128a**  HomeDir=c:\  AlwaysAllowLogin=1  TimeOut=600  Maintenance=System  Access1=C:\|RWAMELCDP |

The MD5 hash of the password is on the line that starts with ‘Password=’.  The two letters after the equal sign are the salt as ASCII text (shown in orange above) followed by the MD5 hash in hex (shown in blue above). This salt is prepended, i.e. the Serv-U password has function is **md5(salt+pass)**.

Your Kali VM comes with a hash cracker called **hashcat**. Read up on how to use it, and have it crack the salted hash above. Tips:

* Hashcat is optimized for GPUs, but your VM doesn’t have one. Override the resulting error message using the --force flag.
* You will need to create a “hash file” as input; the format of this file is “**<hash>:<salt>**”.
* The default brute force mode will be fine, and should take around 30 seconds on the Kali VM.
* Found passwords are echoed in **<hash>:<salt>:<pass>** format and also saved to **~/.hashcat/hashcat.potfile**.

**Paste (1) the command executed, (2) a screenshot of the output, and (3) the password itself below.**

|  |
| --- |
| **OPTIONAL: For up to 6 points of extra credit, crack each of the following Serv-U hashes.**  You may need to use more time, wordlists, and possibly even a combination wordlist/brute-force attack.   1. **qje39945197dbc380a5ac611fbbb7b5354** 2. **csa2eb704ec429728c1fe15814c006a4cc** 3. **qj847d6d5952baef48fa101d84a18b8de8** |

**Question 7: John the Ripper (10 points)**

To start, do some research, and **describe in detail (purpose and content) the /etc/passwd and /etc/shadow files on a Linux system. (4 points)**

For this exercise, we will use [John the Ripper](https://www.openwall.com/john/), which is already installed on your Kali VM. I’ll be supplying the shadow file for the exercise.

We will be using the shadow file here: <http://people.duke.edu/~tkb13/courses/ece590-sec/homework/shadow>

Download this shadow file to your Kali VM. Unlike hashcat, john has pretty smart defaults and wordlists, so you can just run it against the shadow file. To improve performance, you can specify **--fork=<N>** to run *N* instances in parallel; set *N* to the number of CPU cores on your system (you can check with top, cat /proc/cpuinfo, etc.).

Run john on the provided shadow file in a screen session (so you can detach, disconnect, then reattach later). Let it run for at least 24 hours. It is not expected to be able to crack all the passwords, but you should get at least 6.

**Paste the output your results with the following command “john -show shadow”.   
(6 points)**

It should look something like the example below:

$ john -show shadow

abcuser:apple1:15358:0:99999:7:::

defuser:orange:15364:0:99999:7:::

ghiuser:gpgtest:15373:0:99999:7:::

3 password hashes cracked, 25 left

Note:  Here, “abcuser” is the username and “apple1” is the password.

**Question 8: Evaluating MFA approaches (8 points)**

Review [this article](https://krebsonsecurity.com/2018/08/reddit-breach-highlights-limits-of-sms-based-authentication/) on a data theft attack against content aggregator site Reddit.

1. What form of Multi-Factor Authentication (MFA) was deployed at Reddit?
2. What are two ways in which the attacker may have gained access to the second factor?
3. What are two alternative forms of MFA that could be deployed instead?
4. Identify another attack in the last 5 years in which SMS-based MFA was compromised.

**Question 9: SSH with MFA (5 points)**

On your Linux VM, you are going to add some extra protections to the SSH server by adding Multi-Factor Authentication (MFA) to your account.

Some research will tell you how to do this using Google Authenticator or the desktop app Authy.

**Show a screenshot of your SSH login with MFA code.**

**Question 10: Hydra (Online SSH Dictionary Attack) (5 points)**

Hydra is an online network-based dictionary attack tool for SSH and many other protocols.  For this example, the tool will take 4 input values: username file, a password file, target list, and a service (ssh).  With that information, it attempts to perform a network-based authentication to the remote machine(s) on port 22. If it successfully authenticates with the remote machine, it shows the results in the standard output with the username and passwords that worked.

For this exercise, you are going to use Hydra to perform an SSH dictionary attack against a machine specially prepared for this purpose, **target.colab.duke.edu**. Do not use the tool on any other targets!

Implementation note: Because I’ve intentionally put a guessable username/password combo on this machine, I’ve used the [ufw](https://help.ubuntu.com/community/UFW) software firewall (based on Linux [iptables](https://linux.die.net/man/8/iptables)) to restrict SSH access to campus IP ranges.

On your Kali VM, do the following.

**Acquire a password list**

A password list is a text file of likely passwords. There are *many* password lists out there, and Kali Linux ships with many of them pre-installed. For our purposes, we will use the Adobe “top 100” password list, which was derived from [a 2013 breach of 150 million Adobe user accounts](https://www.theverge.com/2013/11/7/5078560/over-150-million-breached-records-from-adobe-hack-surface-online). You can find this password list in **/usr/share/wordlists/metasploit/adobe\_top100\_pass.txt**

**Build user list (1 point)**

If an attacker is doing a general scan, they will typically use a common username list, such as [one of these](https://github.com/insidetrust/statistically-likely-usernames). If an attacker is focused on a known server or organization, they will do research using public information to create a list of likely usernames.

In this scenario, let’s assume we are targeting an organization’s senior leadership, which includes Susan Hypothetical, Scott Samplesberg, and Jacob Exampleface. Based on this, produce a small text file of likely usernames.

**Paste your username list below.**

**Attack (4 points)**

Run hydra against target.colab.duke.edu using the resources gathered.

**Give the hydra command you used below.**

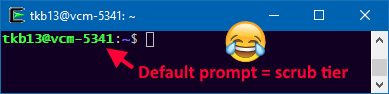
**Display the successful results of your Hydra Attack below.**   If you don’t get any results, try more (and simpler!) usernames.

**When you log into target.colab.duke.edu using the credentials found, a secret message is displayed. What is it?**

|  |
| --- |
| **OPTIONAL: For up to 5 points of extra credit, get a shell on target.colab.duke.edu.**  The vulnerable account on the server is configured to print a secret bit of text and exit without providing a prompt. **Show how you can get an interactive shell using these credentials.** |

**Question 11: Craft your prompt (3 points)**

A rite of passage of UNIX users is the customization of the bash prompt. The default, even on modern Ubuntu, is pretty lame.

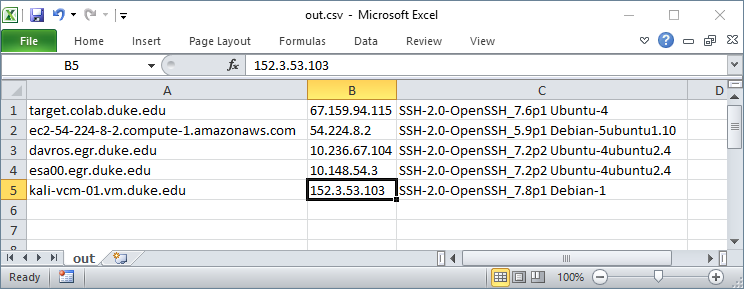


Using your understanding of shell color codes, develop a custom prompt by modifying the PS1 environment variable. You’ll need to mark the escape codes for the shell using \[ and \] indicators (this the shell needs to know what characters are printed vs. interpreted for cursor control purposes); [details are here](https://www.cyberciti.biz/faq/bash-shell-change-the-color-of-my-shell-prompt-under-linux-or-unix/).

**Paste your PS1 environment variable command & a screenshot of the resulting prompt.**

**Question 12: Attack recon (6 points)**

One of the first things an attacker will do when focusing on a particular target organization is conduct reconnaissance. Using your choice of tools we’ve learned, develop a command or script that will produce a CSV file containing the hostname, IP address, and SSH version of all the hosts listed in an input file. Example output (once loaded into Excel):



Apply this to the following hosts:

target.colab.duke.edu

144.202.30.145

155.138.231.109

155.138.195.55

davros.egr.duke.edu

storemaster.egr.duke.edu

kali-vcm-41.vm.duke.edu

(Do not scan machines other than the above. All of the above are machines are administered by me either at Duke or at a cloud hosting provider.)

**Give your script, code, or other artifacts inline in the answer PDF along with a screenshot of it in use. Your solution should be as small and simple as possible!**

**Question 13: Keeping up with the news (10 points)**

One absolutely essential attribute to the field of information security is that **it changes**. While most of the base theory we’re talking about will remain true long into the future, attackers and defenders are locked into a continuous arms race of ever increasing sophistication. As such, a core skill for any security practitioner is to keep up to date with developments in the field.

Below are four IT security news sources:

* [SANS Information Security News](https://isc.sans.edu/newssummary.html)
* [Security Week](https://www.securityweek.com/)
* [ThreatPost](https://threatpost.com)
* [Reddit /r/netsec](https://www.reddit.com/r/netsec/)

Browse each, and pick out an article from the last 30 days related to one of the following topics: networking, cryptography, user authentication, malware, or denial of service attacks.

**From this article:**

* **Provide a brief summary, including a link to the article. (3)**
* **Identify which aspects of the CIA triad are at play and how. (2)**
* **Explain the threat model(s) at play (assets at risk, vulnerability at play, attacker’s capabilities and knowledge). (2)**
* **If your article is explaining an attack, describe a defense that would have mitigated the attack and explain why it would be effective. Use the threat model in your analysis.  
   *-or-*If your article is explaining a defense, describe an attack that the defense would mitigate and explain why it would be effective. Use the threat model in your analysis. (3)**

**Question 14: Manipulating binary file formats (5 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](https://en.wikipedia.org/wiki/Zip_(file_format)#File_headers), the napkin drawings by Julia Wolf in [PoC||GTFO 01:05](https://www.alchemistowl.org/pocorgtfo/pocorgtfo01.pdf) (“This ZIP is also a PDF”), or [this way-way-too-detailed presentation deck](https://www.troopers.de/wp-content/uploads/2011/04/TR11_Wolf_OMG_PDF.pdf) also by Julia Wolf.