University of Chicago
CMSC 23200/33250: Introduction to Computer Security, Winter 2020
Course Staff: David Cash, Alex Hoover, Rohan Kumar, Blase Ur, Valerie Zhao

Assignment 3: The Limits of HTTPS

Due at 11:59pm Thursday, January 30

Introduction

This assignment is very different than the previous two. In this assignment, we intend for you to:

- See firsthand what is happening on a wireless network and evaluate its security implications.
- Gain an intuitive understanding of what HTTPS does and does not protect.
- Learn about side channels, which are ways in which information is leaked through the implementation of a computer system, rather than the conceptual approach.
- Experience the messy nature of real-world data.
- Interface with a widely used computer security software tool.
- Think conceptually about how to exploit an information leak systematically.

Start early! The different steps build directly on each other, so you can't make progress on subsequent parts until you have completed the previous. Crucially, note that Step 1 in this assignment requires that you collect data while physically situated in JCL!

The Broad Setting of this Assignment

"HTTPS traffic is encrypted. Therefore, no one can know what I'm browsing."- A false statement spoken by someone who has not taken CS 232.

The John Crerar Library is haunted! Here in JCL, the Ghost of CS Students Past (subsequently abbreviated *GCSSP*) died an untimely death in JCL one night when the Bomb Project unexpectedly exploded. Some say GCSSP still sits in the halls of JCL all day and night, mindlessly browsing the same website over and over. Thankfully for their privacy, GCSSP took Intro to Security, so they know to browse over HTTPS.

Unfortunately, GCSSP's privacy is not actually safe. Might it still be possible to reconstruct precisely what they browsed even though they're using a version of HTTPS without any known vulnerabilities? The answer is yes, and in this assignment you will do so by exploiting a side channel. At a high level (details follow later in this document), here's what you will do:

1. Step 1: Leveraging software that is widely used in industry, you will snoop on Wi-Fi network traffic to grab the relevant traffic generated by GCSSP, which is mixed in with lots of real traffic from the many networks around JCL.

2. Step 2: You'll comb through this traffic manually and use context clues to figure out which traffic is GCSSP's and gain an initial understanding of what is happening. You'll then start thinking about how you can use side channel information leaked on the network, even for HTTPS traffic, to learn which website they were visiting.

3. Step 3: Now that you know what website they were visiting, can you use additional information leaked in this traffic to determine exactly which page they were visiting even though the traffic is encrypted (using HTTPS)? SIZE UP this traffic (cough cough). In this step, write code to pre-process the traffic you observed. Throw out irrelevant traffic, and associate related flows that originated from visiting a single page. You'll use your pre-processed version of the traffic in this next step.

4. Step 4: Next, you'll need to do some measurement yourself on that website. By cross-referencing your measurements with the pre-processed traffic from the previous step and utlizing the information leak you observed, you'll be able to determine exactly which pages on the website GCSSP visited, in which order. You are permitted to do this step (visiting pages and cross-referencing what you observe with your measurements) either manually or programmatically. In order to make manual analysis tractable, the flag will consist of only the first 10 pages GCSSP visited. This is the only flag in the assignment.

5. Step 5: (Because this last step is a lot of work, we've chosen to make it worth only 5% of your grade for the assignment. In essence, you can skip it and still get up to a 95. However, if you find the previous steps exciting, we encourage you to complete it.) Could you mostly automate Step 4? Write code that automates the measurement and analysis process you used in the previous step. By doing so, you can uncover the full set of over 80 pages that GCCSP visited without having to do the analysis by hand!

Rules

Collaboration policy. Please respect the following collaboration policy: You may discuss problems with up to 3 other students in the class, *but you must write up your own responses and your own code. You should never see your collaborators' writing, code, or the output of their code.* At the beginning of your submission write-up, you must indicate the names of your (1, 2, or 3) collaborators, if any. You may switch groups between assignments but not within the same assignment. As we detail below, you may collaborate more closely on Step 1, potentially sharing the same file that you jointly collect.

Sources. Cite any sources you use. You may Google liberally to learn basic Python and Wireshark and how to use relevant libraries or utilities. You should not Google for anything directly related to solutions to the problems. Searching for posted solutions to similar assignments at other universities is not allowed, although as far as we know this is the first time a class has done an assignment like this.

Campuswire. We encourage you to post questions on Campuswire, but do not include any of your code in public Campuswire posts. If you have a question that you believe will reveal secrets you have discovered while working on the assignment, post privately to just the instructors. If you have a question that you believe will be of general interest or clarifies the assignment, please post publicly. If you are uncertain, post privately; we will make public posts that we believe are of general interest.

Outside attacks. Your attacks for this assignment should be those discussed below. Do not attempt to compromise our server, sniff your classmates' network traffic, or do other nefarious

things. You will not receive credit for breaking into the server. In fact, you will lose credit for doing so.

Grading. You will be submitting data you collect, code you write to process that data, a single flag for the whole assignment, and explanations of many (but not all) of the steps. As we detail each step, we explain what code (if any) and data collected/created (if any) to submit for that step, as well as the file naming convention. Please submit your explanations and the flag as a single PDF file with the answers corresponding to the different steps marked very clearly. Responses will be graded for correctness and clarity. Point values are given below.

Assignment Tech Set-Up and Overview

We wrote our reference solution in Python 3. While we will be able to provide the most comprehensive support for doing this assignment in Python, you ultimately may write code in whatever language you feel most comfortable with. Just make sure that your language of choice has a good library for parsing a JSON file.

Accessing the server

For this assignment, the website GCSSP is visiting is accessible on a public IP. There's no need to be on the UChicago network or use a VPN. You must physically be in Crerar for Step 1, though, because of the nature of the task.

How to Submit

Upload the files specified in each step to Canvas: (`https://canvas.uchicago.edu/courses/25992`).

Step 1: Getting the Network Traffic (25 points)

GCSSP just happens to be browsing the Internet right near David Cash's office. Therefore, to capture GCSSP's network traffic, you'll need to hang out near David's office (JCL 353). We suggest setting up shop in the lounge area next door to David's office. Please don't sit in the hallway in front of his office!

Well, how do you capture network traffic? We strongly recommend using Wireshark, which is a free and open-source tool for analyzing network traffic. To be pedantic, "Wireshark" refers only to the tool for analyzing traffic. That said, installing it also installs the software that actually captures traffic from the network, and Wireshark is a nice wrapper for that software. Wireshark can be downloaded from `https://www.wireshark.org/` or your favorite package manager (e.g., apt on Ubuntu). Note that you might run into problems installing naively via brew on a Mac, though some quick web searches should point you in the right direction. For Macs, it is realistically easier to just download Wireshark from the website.

Before we go further, **some very important caveats**. If you're running Windows (especially Windows 10), Wireshark probably won't do what you need because most drivers on Windows do not permit you to enter monitor mode. If you're running Windows, you therefore might consider making

a bootable flash drive of Kali Linux, a security-focused distribution that comes with Wireshark pre-installed. Furthermore, even on machines running Mac and Linux, some network cards will not work or not support what you need. For example, we learned this week that a brand new Mac running the newest Mac OS wouldn't work. Because we don't want this step to frustrate you for many hours, we've taken the following steps to support the process if your personal hardware doesn't work with Wireshark. First, Alex has an extra Mac laptop and Blase has an extra Ubuntu laptop (both confirmed to work for this task) that we are happy to lend out for you to perform these measurements. Check with us on Campuswire to set a time to borrow one of these devices. Second, we aren't requiring every student to create their own Wireshark capture. You may work with up to three other students (see the collaboration policy above) on the conceptual parts of the assignment, as usual. Furthermore, you can collaborate even more closely on Step 1. You may do this step together, sharing the same laptop. This means you will all share the same capture file. Our requirement, though, is that you *all must be present and engaged while capturing the traffic, even if you're only using a single person's laptop to do so.* Finally, if all else fails and you're still stuck on this step after about 2 hours, please post on Campuswire and we'll work with you to help. We think it's instructive for you to get the real-world experience of capturing traffic. Of course, doing anything with computers in the real world can be somewhat frustrating, and we of course want to limit the frustration to reasonable levels in the context of a graded assignment.

Some good things to know before starting:

- GCSSP is connected to the "securityclass" network, which you'll notice isn't using any Wi-Fi encryption. This makes the capture a bit easier, though note that Wireshark can still decrypt traffic if you know the password to a WEP/WPA/WPA2 network. As we'll discuss later in the course, you can sometimes break into Wi-Fi traffic even if you don't know the password, particularly for WEP.
- You can connect to the "securityclass" network, but it is filtering by MAC address (and yours isn't on the list of permitted MAC addresses), so you won't be able to connect to the Internet on this network. Connecting to the network, though, helps convince your Wi-Fi card to use the right settings in subsequent steps.
- You should set your Wi-Fi card to **monitor mode** (see below). Monitor mode lets you capture all Wi-Fi traffic around you without needing to connect to any network. In contrast, **promiscuous mode** tells your network card that you want to see all traffic (either on a network you're already connected to or all traffic observed in monitor mode) regardless of whether or not the traffic is intended for you.
- Wi-Fi networks operate on different channels (small frequency bands). In general, you need to listen on a particular channel even in monitor mode. The "securityclass" network is a 2.4 GHz network running **on Channel 6**. You may need to explicitly set your network card to listen on Channel 6 if you aren't seeing the expected traffic.
- The order of steps matters, and sometimes the proper order depends on your hardware. For instance, to switch to monitor mode, some network cards prefer not to be connected to any network at the time. If you enter monitor mode in Wireshark and are seeing no traffic, try disconnecting from all networks, turning on monitor mode, and then connecting to the network.

Given those caveats, here's what you should do. First, we'll give the abstract instructions. Then, we'll give specific examples that will hopefully apply to many Mac, Ubuntu, and maybe even Windows setups. Install Wireshark. Connect to the "securityclass" network (at which point you'll

lose your typical Internet access). Then, you'll need to set your Wi-Fi card to both **promiscuous mode** and **monitor mode** (potentially specifying Channel 6) using either the Wireshark interface or command line magic. Once that happens, start capturing traffic in Wireshark. You should see traffic start appearing quickly from lots of networks. All relevant traffic will be on the "securityclass" network, but it may not always be labeled as such. Note that when you're in monitor mode, you won't have your normal Wi-Fi access. It'll be good to have a smartphone or other device around to look up instructions or help while your computer is in monitor mode.

Approximate detailed instructions for Mac OS. Note that if your hardware is too new, it's possible that this won't work.

- Download wireshark from `wireshark.org`
- Make sure to drag the application into your folder and install ChmodBPF
- Before opening Wireshark disconnect from whatever Wi-fi network you're on and instead connect to "securityclass". This is the easiest way to force your Wi-Fi radio to Channel 6.
- Open Wireshark and switch off all interfaces except for Wi-Fi, which is typically named "en0" on Mac laptops
- Click the setting gear in the top left
- On the *Wi-Fi: en0* line, check the "monitor" box, also verify that "promiscuous" is checked.
- Go to Wireshark's main page and begin capturing traffic. See the shared instructions below to start making sense of what you see.

Approximate detailed instructions for Ubuntu 18.04.

- Nearly all of the commands on the command line require "sudo", so I just ran "sudo su" to switch to a root shell for the whole adventure. Note that Wireshark does not need to run as sudo, and I opened it through Ubuntu's GUI anyway.
- On the command line, run "apt-get install wireshark"
- Connect to the "securityclass" network to try to convince your Wi-Fi radio to be on Channel 6.
- Next, you need to know the name of your Wi-Fi card. On the command line, run "ifconfig" and figure out through context clues which is your adapter. Names like "wlan0" are common. On Blase's laptop, it was called "wlp3s0" for some reason. Replace all instance of "wlp3s0" below with the name of your adapter.
- Run "service network-manager stop" to stop Ubuntu's network manager from controlling your card. At this point, you've probably been disconnected from your Wi-Fi network.
- Run "iwconfig wlp3s0 channel 06" to set the channel to 6.
- Run "ifconfig wlp3s0 down" to turn the card off temporarily so that you can change its mode.
- Run "iwconfig wlp3s0 mode monitor" to change to monitor mode. You can verify this by running "iwconfig wlp3s0" and seeing the mode it shows.
- Run "ifconfig wlp3s0 up" to turn the card back on.
- Run "iwconfig wlp3s0 channel 06" to convince the card it really should be on Channel 6.
- Open Wireshark, verify that both "promiscuous" and "monitor" are checked for "wlp3s0" on the "Capture" $\rightarrow$"Options" menu and that you have selected "wlp3s0" as your device. Then, start your capture. See the shared instructions below to start making sense of what you see.
- After finishing the capture, I used the following commands to get my computer back to normal (though rebooting also works): "ifconfig wlp3s0 down"; "iwconfig wlp3s0 mode managed"; "ifconfig wlp3s0 up"; "iwconfig wlp3s0 channel auto"; "service network-manager start". If the Wi-Fi was still being weird, I used the network manager to turn off and then on my Wi-FI.

Approximate detailed instructions for Windows. Note that many network cards, especially on recent versions of Windows, do not support this.

- Download wireshark from `wireshark.org`
- Go through the install, and make sure to install **Npcap** because Winpcap typically does not support monitor mode.
- Connect to the "securityclass" network to try to convince your Wi-Fi radio to be on Channel 6.
- Open Wireshark and uncheck all non-wireless interfaces
- Select both "promiscuous" and "monitor" are checked for your Wi-Fi card.
- Begin capturing packets. See the shared instructions below to start making sense of what you see.

Some helpful hints:

- GCSSP is somewhat neurotic. They start their browsing over and over again every 15 minutes (at 0, 15, 30, and 45 after the hour), 24/7. There's some randomness involved in their browsing session, but it typically lasts 5–10 minutes. Their first request is even over HTTP, not HTTPS!
- GCSSP has the private IP address **192.168.1.2** assigned on the "securityclass" network. You can assume that they are visiting a **website hosted on a UChicago IP address**. In Wireshark, you can use the filter command `ip.addr == 192.168.1.2` to look only at traffic related to GCSSP's computer. As this is a computer running a real operating system, you'll notice some auto update requests and other incidental traffic that is not relevant to the assignment.
- If you find traffic that matches these assumptions and seems to be making a small series of requests every few seconds, you've found the right stuff! Note that since GCSSP's browsing repeats every 15 minutes, as long as you start capturing by the start of one of these sesssions, you need at most 15 minutes of traffic. After you observe 30+ seconds of network silence from GCSSP, you may assume their cycle is done and that they are waiting to repeat their browsing during the next time window.

After you are done capturing the packets, we highly suggest exporting them as JSON. While there are Python libraries for dealing with Wireshark's default pcap (packet capture) files, we have found them to be buggy. Go to File→Export Packet Dissections→As JSON to save a JSON file. Note that you may filter out some of the irrelevant traffic using Wireshark, though you don't need to. In any case, expect this JSON file to be large (on the order of hundreds of megabytes).

What to submit. Upload the file with your packet captures as JSON (or, if you really prefer, as a pcap file). Assuming the former, name this file "capture-CNETS.json", where CNETS is replaced with a hyphen-delimited list of the CNet IDs of the students sharing this file. For example, if Blase and David collaboratively captured this trace, the file would be called "capture-blase-davidcash.json". Note that this file might be hundreds of megabytes large.

Step 2: Examining What The Traffic Contains (10 points)

Now you need to start manually making sense of what you captured. We suggest you use Wireshark since the filters and graphical interface will likely help you comb through this traffic. First, find the traffic that pertains to GCSSP visiting a website with an IP address on the UChicago network.

What website are they visiting (not just the IP address, but also the host name)? There are a few different ways for you to determine this; you'll have to figure out at least one of them.

What to submit. You start the writeup with this step. In your writeup, include a section called "Step 2" that describes in about two paragraphs what you observe. What's contained in this traffic overall, as far as you can tell? Feel encouraged to draw heavily on the material Blase covered in lecture. Most importantly, what security-relevant information is observable by anyone sniffing the network traffic based on what you see? Finally, include a third paragraph specifically focused on what GCSSP is doing. What website are they visiting? List both the IP address and host name.

Step 3: Pre-Processing and Filtering the Network Data (30 points)

Now that you know what website they were visiting, can you use additional information leaked in this traffic to determine what page they were visiting? As we hinted before, maybe the number and size of data flows your observe could be useful here. As you'll observe, visits to a page often result in multiple rows of data in Wireshark (and in the JSON trace). In this step, we want to get rid of irrelevant network data and also as much of the irrelevant parts of GCSSP's own browsing as we can.

In this step, write code in whatever language you prefer to pre-process the traffic you observed. Throw out irrelevant traffic, and associate related flows that originated from visiting a single page on the website. You'll use your pre-processed version of the traffic in the next step.

What to submit. First, in your writeup, include a section called "Step 3" that describes in about a paragraph what information you decided to keep and why you decided to keep that information. Second, also submit all code you wrote for this part. Please use the prefix "step3" with whatever suffixes you want (e.g., "step3CleaningCode.py"). Third, also submit the processed output of this step in a human-intelligible format that makes sense for your next step (e.g., perhaps a text file or a JSON file). Please use the prefix "step3result" with whatever file extension corresponds to your decision (e.g., "step3result.json").

Step 4: Measuring the Website (30 points)

Your goal in this step is to determine the first 10 pages GCSSP visited on the website, in order. This will be your flag in this assignment.

From Step 2, you know what website GCSSP was visiting. From Step 3, you have pre-processed data of what GCSSP's page visits looked like in terms of the network traffic each created. How might you determine which page GCSSP was visiting? Well, try to retrace GCSSP's steps yourself and see what network traffic that page visit generates. Fire up Wireshark on your own computer and see what happens when you visit each page linked from index.html on the website GCSSP was visiting. In this case, you only care about your own traffic. You don't need to be in promiscuous mode or monitor mode. In fact, it will probably be easiest not to be in either. (Plausibly, you could instead use a tool like wget to download pages from the website and measure them. While that is an acceptable solution, examining traffic through Wireshark will be easier.)

Important hints and assumptions follow:

- While looking through the network traffic in the previous step, you hopefully have noticed that GCSSP started their browsing on index.html. That's their starting point, though don't

count this as one of the 10 pages that make up your flag.

- **Assume that all of GCSSP's subsequent browsing came only from clicking on links.** They didn't type anything into the URL bar, hit the back button, or anything like that. In other words, at each step, there are only a few possible next steps.
- Assume also that pages on the website GCSSP is visiting are static. They don't change over time.

You might notice that these are some strong assumptions that will make your task much, much easier. That said, research over the past decade has shown that the same sort of techniques you are using can also be adjusted for cases with far weaker assumptions.

What to submit. In your writeup, first include a section called "Flag" that contains the first ten pages GCSSP visited after index.html. List these one page per line. Include only the name of the html file of the page, and exclude the file extension. For example, if `https://www.example.com/12345.html` was visited, list "12345". Second, include in your writeup a section called "Section 4" and describe your approach to this problem in two or three paragraphs. As part of this description, also discuss problems that your approach will face in the real world, alongside explaining the intuition for how you might overcome those difficulties.

Step 5: Automating the Attack (5 points)

Because this last step is a lot of work, we've chosen to make it worth only 5% of your grade for the assignment. In essence, you can skip it and still get a 95. However, if you found the previous steps exciting, we encourage you to complete it.

In the previous step, you (probably) manually browsed the website GCSSP was visiting to try and understand what they did at each step in the browsing. However, an alternate approach would be to write code that crawls the website, creating a representation of the possible paths and what leaked information would suggest each step in the path. To that end, write this code. We highly recommend Selenium or other browser automation software.

After running this measurement step, you should save your representation of the website and the information each potential step in the path leaks. You should then write additional code that takes two inputs. First, it takes in pre-processed data of network observations of GCSSP (in whatever form you ended up with at the end of Step 3). Second, it takes in the representation you just created. It will then output the full set of over 80 pages that GCCSP visited without having to do the analysis by hand!

What to submit. Submit all code you wrote for this step, as well as the two files you take as input (your pre-processed data and your representation of the website). Prefix all files with "step5", using whatever suffixes you want. In addition, include in your write-up a clearly marked "Step 5" where you explain your approach in a paragraph or two. Finally, include a section called "Full Flag" in your writeup with the full path GCSSP using the same format as in the previous step.