Chris Cohen

HONEYPOTS

Creation, Deployment, and Analysis

Purdue University

CS422 - Final Project

April 28, 2020

1 Problem Definition

A honeypot is a term used in the cyber security field describing a network entity that purposefully lures attackers. It is particularly useful for frustrating and distracting attackers, since it is typically a dead end with no access to any real, desirable information. Attackers are constantly scanning the internet for vulnerabilities, and can potentially shut down an entire network if given the opportunity. A honeypot is an extra layer of security for a network, and even though it may not capture every attacker, it can take a good portion of malicious traffic away. One common use case for a honeypot is to set up some sort of monitoring, and collect information on any attackers. Analyzing this information can be valuable to help improve network security in other places that need it.

READ ARTICLES AND PUT MORE HERE

2 Motivation

I chose this project in order to learn more about my field of study, cyber security. As I've been applying to internships over the past year it's been increasingly evident to me that this field is tough to break into. When a company looks for an employee to help secure their important assets, an inexperienced student is their last choice. Employers think highly of students who have completed personal projects, so this was the perfect opportunity for me 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 caught my eye because I had never heard the term before. The idea intrigued me - it seems so simple, yet effective if done right. I already had a simple HTTP web server from the first lab to build off of, so I wouldn't be spending much time on the basics. Additionally, in the future, I could convert it into a normal web server for my personal needs. Going into this project, I realized that this would

be a challenge, as hosting a web server on your home network isn't advised. Exposing your network to the outside world will certainly pose a threat if done improperly. This ended up being a blessing in disguise, as it helped me learn much more about network security than I anticipated.

3 Related Work

3.1 HoneyHTTPD

One particular project that inspired me is *HoneyHTTPD*. The idea of this project is that it 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 SSL certificates for the fake web server. Hartman [1] 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 for a potential attacker 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, a Raspberry Pi 3 B+, suffered in performance. Overall, this project was a very good start for brainstorming, and gave me multiple good leads to start testing my honeypot.

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", my first thought was to use one for spam detection and prevention. 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 exact idea, 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 IP associated 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.

4 Completed Work

PUT WORK HERE INTEGRATED WITH SCHEDULE

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 more problems than anticipated. The base code was completed by Saturday, as planned, but deploying it took a bit of work, as mentioned in the *Proposed Work* section above. After I ironed out issues such as SSL certification and redirection to force an SSL connection, the initial test run was deployed on Sunday, April 26th. After encountering multiple attacks while hosting on my home network, I decided instead to host on Amazon Lightsail, a service that provides a VPS [1]. I chose this because it offered the first month free of charge, and gave me a full-fledged virtual Ubuntu 18.04 LTS server to work with, nearly identical to my home setup. After successfully setting up the VPS, I collected data for the rest of the week. I started out with my originally planned HTTPS server, but I noticed in the active server output that multiple clients attempted to connect with HTTP, were rejected, and never attempted to reconnect. Given the short period

[1] Virtual Private Server

of time that I had to test, I wanted as many data points as possible. I saw no drawback in only using HTTP, especially since I'm not sending anything important or hosting the website on my own network. From Wednesday, April 29th and onwards, I solely used HTTP to serve requests, giving me more data points to work with.

5 Conclusion

References

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

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