SOS Project Reports
Skid2.0
Evergreen CTF 2018

Andrew Jordan, Alexander King, Kreig Clemens, and Jacob Golding

The Evergreen State College,
Olympia, Washington
{jorand25,kinale26,clekre03,goljac24}@evergreen.edu
http://ada.evergreen.edu/sos

Abstract. This paper combines multiple resources and challenges that will help the reader's ability to both understand our way of creating the challenges and to solve them.It dives into web security challenges, and what we learned from reading up on, and creating our own. The goal of our project was to create and host web security and various other challenges on our updated website.

PHP, MySql, JavaScript, Hacking, Challenges, Skid2.0, Music4Robots, Resources, Web Security

1 Introduction

This project is guided as a compilation of hacking challenges, on a new website and to update the website to a more modern view. Challenges are broken up into, people who created them and what challenges they did. For Andrew, it was building a website challenge, that incorporates both web-based and cryptography focused challenges. Alexander had build a buffer overflow challenge that you would download from the website and would be able to run the C code on your own computer or practice it on a page of the website. Kreig worked on SQL and PHP penetration. First creating a local server to test the querying and then shifting it over to the dedicated website with its own querying in place. Jacob was tasked for porting over and updating the previous site that hosted hacking challenges. Changing and updating a previous website that was used for previous challenges that all the members have done before in some way or another.

1.1 History/Background

Originally the project was set to be a interesting twist on hacking challenges, known as CTF's. The group wanted to bring the livelihood of downtown Olympia and the enjoyable processes of hunting down flags in a real world situation. After a few weeks of planning, marking out all of the projects and asking local businesses downtown to hold a physical object that can lead to challenges, we

found ourselves a little ahead of ourselves. After focusing on what we wanted to do, we we're going to host our project with the assistance of Spring Science Carnival. Sending an email and request of data, signing some papers and getting ahold of staff who run the carnival, we got an email back that did not sound like this was a proper fit for our project, so we tried getting ahold of a few other people for new ideas. We talked and tried to just focus on our work, and if students after our presentation would want to play the challenges, we would be able let them connect to the website and try the challenges themselves.

2 Specification of the Project

The specification of the project is in multiple parts, most of the challenges are built from multiple parts. Are challenges span from multiple parts, ranging from HTML/CSS/Bootstrap/MySQL for the web based challenges, C and x82 instructions for the buffer overflow, MongoDB and Googles hosting platform for the cryptography challenges. We followed a general outline of any CTF based website, you take common computer science vulnerabilities and create a problem that could produce a flag for solving or completing the challenge. We all had an interest in the challenges we created because they were the ones we worked on the most, meaning we would have an already set knowledge and outcome of how we wanted them to turn out. Our final outcome would determine success, if we felt like they were interesting, difficult in the way we wanted them, and made the user interested enough to go out and seek more knowledge about the subject afterwords.

2.1 Use cases

The use cases for this project is for people who want to learn more about a problem in computer science and want to explore that area by solving a challenge. We also have connected other resources, the ones we looked up to help guide us, so if the user have further requests or are stuck on a challenge you have the resources to do so.

2.2 Implementation choices

For implementation, we first start off with a foundation of an updated website. It was built on PHP, Bootstrap, JavaScript, and MySql connected through a hosting application MAMP. Other challenges include, hosting an entire website full of challenges, which is hosted on the Google Cloud. Alex's buffer overflow challenge is a file hosted on the website that you can download and work on through the steps provided. Finally, Kreig's is hosted on Skid2.0, implemented the same way the other web based challenges are, through the browser themselves and modifying the database, different then the one hosting user data.

2.3 Software Architecture

For hosting the servers, we wanted to have a dedicated server to host both Skid2.0 and Music4Robots. During the project we worked mostly on our own computers, for building the challengers. Skid2.0 was mainly built on a PHP/HTML/CSS, along side the server tool MAMP to connect the website, and to connect to the database is a package that comes with PHP, PhpMyAdmin. These tools all worked in tandem with connecting them all together and even hosting them through a port on a local network that would allow users to pull data from Jacobs computer. We didn't want to host off our computers for reasons that happened during the presentation, we were trying to show off the website, Jacobs computer turned to sleep mode and then no one could connect.
For our other website that is hosted on Google's platform, we wanted to use another platform that wasn't directly hosted from a computer, so using this was a great secondary case. Andrew used this for his challenge set, and it was used with HTML/CSS and MongoDB. The only downside to this, is you have to pay for anything greater than 14 days, so needless to say, we had to commit it to being up and running 2 weeks before the end of our presentation. It worked seamlessly, it hosted most of the challenges fine, the only problem is that sometimes when people connect to the website, a challenge that would play on the main page, was a cryptography challenge based on sound.

2.4 Test Cases and Comments

In this section, everyone is going to talk about their projects, seeing as this is a 4 person project, and most of the general information summed above in general for the project, and the use cases.

Setting up the website - Jacob In redesigning the skidctf website it was necessary to learn php scripting language. The site used php to query the database, and to check for the status of logged in users. In addition, I wanted the site to look more profession and be more responsive. To accomplish this goal, I used the front-end framework of bootstrap. With this I was able to standardize and sharpen the look of the website as well as give it more responsive functionality. Finally, to host the website I chose to use a MAMP server.
The biggest challenge is redesigning an older website came from the changes in method that come with the newer version of php. All the queries to the database used an outdated method call. It took some time reading the documentation to learn about this issue. Another big issue came from the fact that my file system was different to the one where the site was hosted previously. This caused all of the 'a' tags to point to invalid location. I had to study the structure and check every link to ensure structure was returned.

Music4Robots - Andrew For the project, I wanted to work on hosting and learning the tools in the cloud. The website music4robots.co was designed with

many security vulnerabilities including; insecure directories with hidden files, stenographic messages in the background gif, audio messages, and exploitable binaries. This was fun and really turned me on to many of the cool tools available on the GCP platform. Since the site was hosted live on the internet, we got to examine the effects of automated traffic on our site. This was staggering and within a day required me to start working on mitigation methods. (fail2ban, firewall rules, etc) The idea behind the site was to create a sort of multi-challenge, that wasn't just one flag.
The other interesting thing was hosting the binary challenges with socat. Alex wrote some cool binary challenges that would take arguments thru argv and print to stderr when the usage was askew. This was not the best way to communicate over a socket. Therefore we needed to change the code a bit and learn some things about how programs will buffer to stdout. Once we disabled buffering and rewrote the code to accept arguments from scanf, things got a lot smoother.

Buffer Overflow - Alex My work over the course of the project involved copious amounts of reading and preparation prior to implementation of my final challenges. Though I have been interested in security and exploitations ever since I was a teenager, I had never actively pursued learning about binary exploitation until the winter quarter of 2018. During winter quarter I purchased Hacking: The art of Exploitation by Jon Erickson and began reading as much as I could to get a better understanding of x86 architecture. With the help of David Weinman and solving many other capture the flag challenges I felt confident enough in my knowledge of C, x86 architecture, and common exploits to begin working on them myself by spring quarter.
Along the way many problems arose. Originally, during winter quarter I was working on simpler projects for the Science Carnival that could be solved by young adults with minor, if not no computer security knowledge. When the Science Carnival was no longer the direction our project was heading I had to start on all new projects more geared towards undergraduates focusing in computer security and was not fully equipped with the knowledge to make this problems until later in the year.

Sql Injection - Kreig Working on the project from the begging, I also ran into problems with starting challenges for the Science Carnival, I spend a good amount of time trying to build and work on projects that would work for that situation but not the web-based challenges I would work on. After switching to the final version of our project and switching to SQL / PHP, I didn't really have a base of how to "hack" a database. I knew how to set up a MySQL server to run and connect on a windows machine that wasn't web based, so I had to start from there. Working on a low level CTF from the previous website that we updated. After having a few discussions with one of the networking teacher assistants, it finally clicked for me and I started getting a better understanding of it. As Well as starting on my own php/sql server, starting from downloading the package and install it on linux. It was simple for the most part, the dependencies

were the hardest to work with though, they have some serious out of date pages and information. Afterwords, connecting the database was easy though, using MyPhpAdmin, the databases were very easy to create and manipulate data, giving me a better understanding of reading the previous websites querying tools.
After completing a few challenges, I started to work on my own, after looking at a few resources from W3 school and how PHP queries the sql, I pieced together a combination of check statements for the first wall, blocking out some of the most fundamental solutions that I used. First I check if it even uses any quotation marks, equal signs, the two largest ways to turn the sql statement true. After checking that I would break the user name and password into two variables, sanitize them, meaning they had to be strings or ints for each char, and then to check if they were inside the database. Hopefully you are not someone who wants to do the challenge because this is a big hint.

3 Status

Most parts of the project is working. The Skid2.0 is up and running, the challenges have been added, the flags that give the completion of the project is working. The database for adding users, the score board and other functionality is working as well. Andrews hacking website, Music4Robots is working for the , but will most likely not be there to check later because of a temporarily hosting issue, as Google only allows you 14 days free on their platform. On the not so working side, the database connected to the web-based challenges are not querying correctly when working with Skid2.0, but locally work fine, we believe to know the solution and are working on it currently

4 Conclusions/Critique

All of the work we set out to do is finished, after changing the project 3 times and losing a few members of the group, we finished gracefully. We all picked a subject that we enjoyed studying and dived deeper into those topics to create challenges for other like minded people who enjoy the challenge of learning. At the end of this project, we all come out a little further ahead, doing the work we enjoyed studying and playing with while doing CTF's. Now it's the users turn to hear the call and complete our challenges, and maybe you solve them, and understand the challenge and interest of our project and create your own.

5 The References Section

References

1. "SQL Injection." W3Schools Online Web Tutorials, `https://www.w3schools.com/sql/sql_injection.asp`

2. DigitalOcean. “How To Install Linux, Apache, MySQL, PHP (LAMP) Stack on Ubuntu 14.04 — DigitalOcean.” 5 Common Server Setups For Your Web Application — DigitalOcean, DigitalOcean, 13 Oct. 2016, https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-14-04
3. Crew., HackThisSite.org. “Hack This Site!” Hack This Site, www.hackthissite.org/articles/read/855
4. “ SKIDCTF RELOADED.” Skidctf Reloaded, ctf.hackevergreen.org/
5. “MUSIC FOR ROBOTS.” Music for Robots, music4robots.co/
6. For more resources for general hacking and programming knowledge, https://evergreenctf.wordpress.com/bibliography/