Ira A. Fulton Schools of Engineering
Arizona State University

CSE 545: Software Security

PCTF Final Team Report

Team Name

Bindevil

Team Members

- Captain: Vo Minh Duy, vduy1@asu.edu
- Kosuke Nagae, knagae@asu.edu
- Hongbo Wu, hongbowu@asu.edu
- He Jiang, hjiang81@asu.edu
- Jinyi Yu, jinyiyu@asu.edu

Part 1 - Project Description and Functionality

How does your team's project function?

As a team, we Bindevil worked together by various methods:

- Meeting by Zoom at weekends
- Chatting and sharing information via our Slack channel
- Sharing files through google drive and GitHub repository

Our plan participating in the game was every team member would analyze the services and perform exploitation/patching based on but not limited to their chosen topic. Because each team member comes with a different background and PCTF covers a wide range of topics from network attacking to binary/web exploitation, each team member picked up topics that he would like to research deeply, as we proposed in our project proposal. We did not split based on roles like offense or defense, but instead, we worked as a whole during the game. We used Zoom and Slack to share our findings during the game, and if one member got stuck or needed to off for a few hours, others would cover his parts while the game went on.

Before participating in the game, we determined that the class's assignments, especially binary and web hacking, provided essential knowledge and practical experiences for the game. While working on these assignments would also help us familiarize ourselves with exploitation tools such as pwn, ghidra, metasploit... So, we tried to finish all these assignments before the game started. Next, we would need a few automated tools to aid us in offense and defense based on the class's different topics. Since there are some vulnerabilities and common techniques to exploit these vulnerabilities, tools were written to automate them.

- **Firewall:**

  After studying the ictf infrastructure, we found that it should be an effective method to apply in both defense and attack if we can build a firewall to handle the traffic routed via our VM. As we have done reflector assignment, we tried to use sniff and duplicate packets, but after doing some research, we found that we could not control the packet in the sniff call back function. Instead of using sniff, we use nfqueue to handle the PREROUTE traffic, which allows our program to act as a firewall. Below is the basic process of the packet flow (Figure 1).

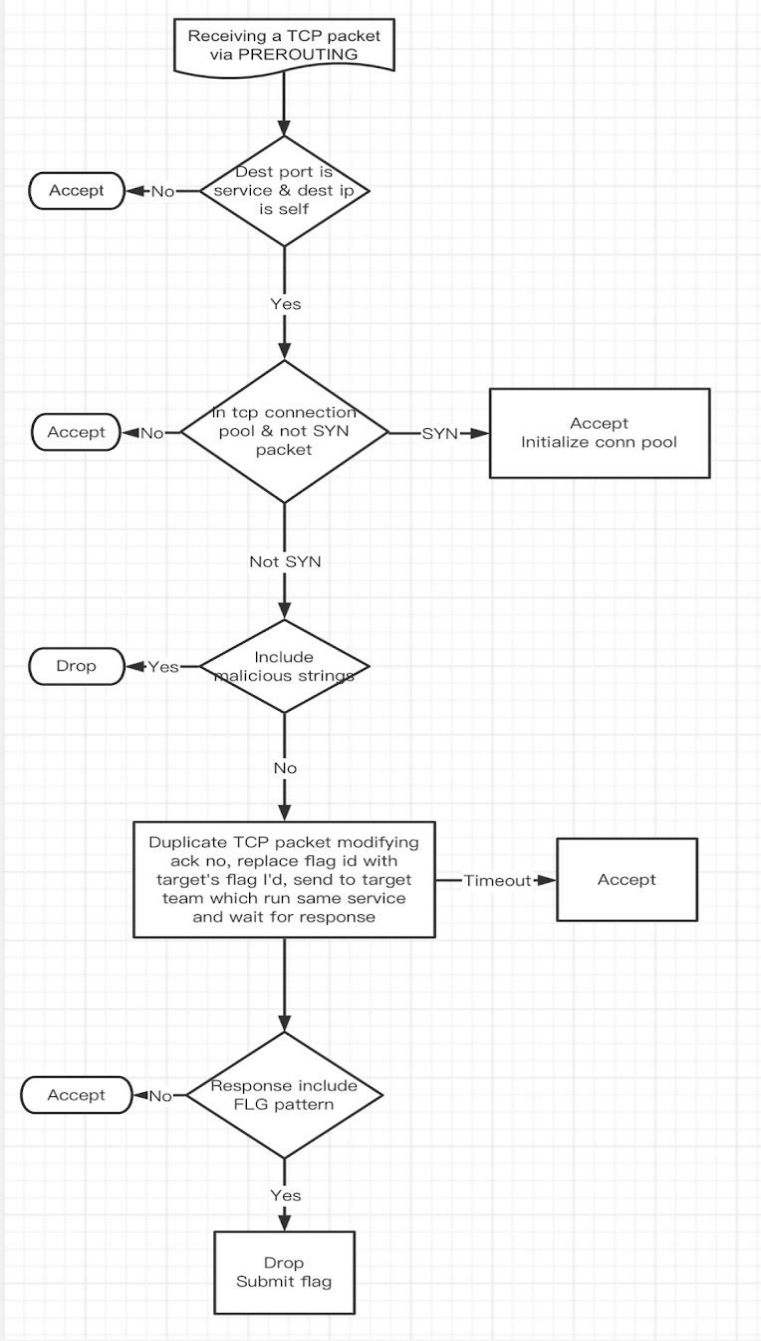

Figure1: Basic process flow of firewall

The main idea of the firewall is to check every TCP packet routed through our VM, drop all the malicious packets, and accept only good ones. Our firewall handled all traffic via our host to the destination port of four services, and this is achieved by setting specific iptable rules. We also need to keep the kernel to respond with the RST packet to avoid affecting the TCP connection to the target team. Iptable rules setting as in Figure 2:

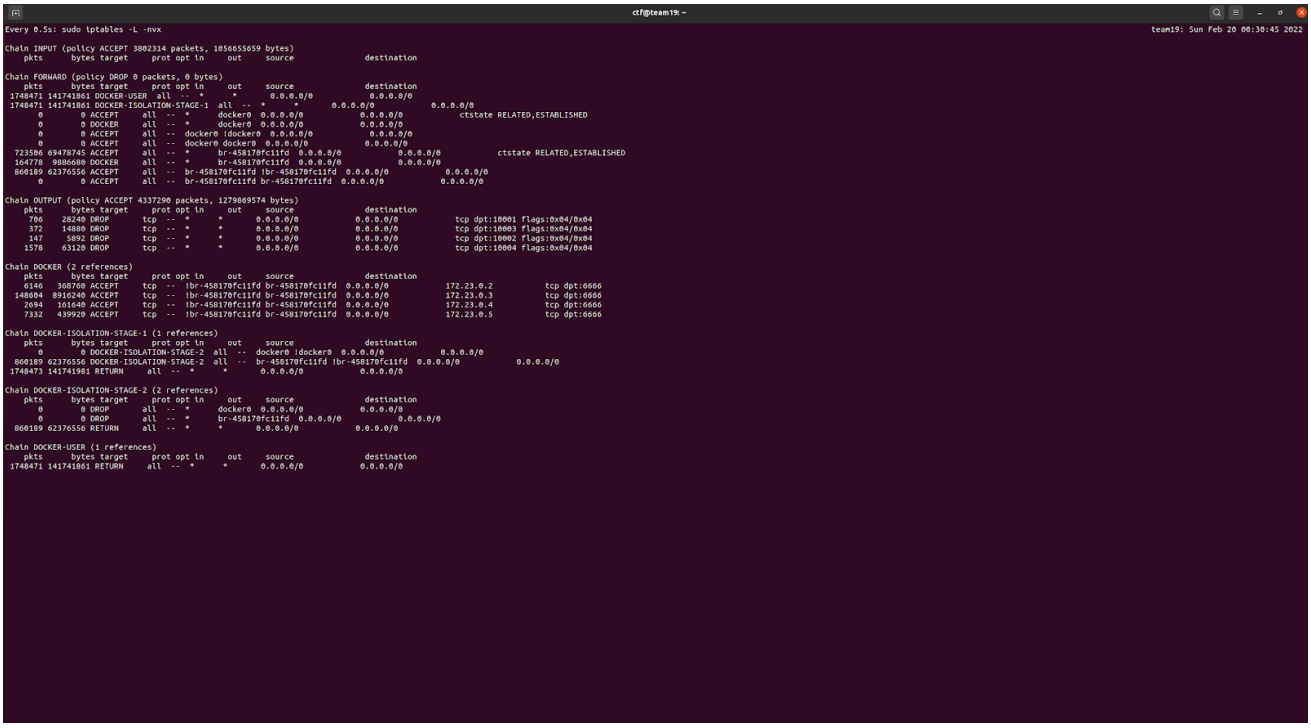

Figure2: Iptable rules

Then we have two ways to determine if a packet is good or not. First, we have a string filter to filter these bad packets. We can also duplicate the received packet and send it to a target team that runs the same service and sees their response with a flag. If they respond with a flag, we will submit the flag. This second way is much more complicated, as we need to control the sequence number and acknowledge number during the whole TCP connection. We successfully make it available for the handshake process and the first few data transfer processes. However, we lose control in the following period when the target has some negotiation packet. We should fix this or apply the strategy in the application layer, such as using the nc command to make the attack more manageable.

- **Vulnerabilities scan script:**

  This python script was created in an attempt to scan suspicious function calls in the source codes. The script finds any function calls that can cause vulnerabilities, such as strcpy in C, include in PHP, etc. Finding this would help us narrow down the vulnerability location, faster patch our services, and exploit other teams' services.

- **Flag submission:**

  The author's idea was to provide a tool with easily remembered arguments for getting the game information like running services, participating targets, submitting individual flags, or a list of flags stored in text files. A tool is built as the wrapper of team interface API using Python language. This tool can be executed in the command lines.

- **Auto exploit script:**

  This is a python script template written for auto exploit. The script uses pwn to connect and send data to the target machines. There are three basic steps as follow:

  1. Use team interface API to get a list of target machines.
  2. For each target, use pwn to connect and send crafted data to exploit the service.
  3. If the exploitation is successful, retrieve and process data, then automatically send the obtained flag to the team server through team interface API.

  When the game started, we processed to exploit the services manually. After successfully exploiting the services, we modified the template script based on each service and used cron job to run exploitation every 3 minutes automatically.

Part 2 - Project Performance Expectations

How and why did your team expect your team's project to help you succeed in the PCTF?

We wanted to cover as many exploits we've learned in this class from the beginning. We expected that PCTF would be involved a lot in binary/web hacking, so we prioritized the binary/web hacking assignment to gain knowledge and experiences as much as possible to prepare for the actual game. This knowledge and experience aided us well during the game as we could use pwn and ghidra to analyze the services at ease. It also speeded up the time to craft and deploy the attack.

We thought that the services' code would be changed/updated/inserted during the game, so we expected the vulnerabilities scanning script would also help us save time to narrow down the vulnerability codes by scanning the common vulnerability function, thus reduce the workload.

We chose to use a firewall because we don't know what service will come in the pctf competition, and the firewall can be used for general protection and attacks. And we don't need to write the code within 24 hours during the pctf time. Therefore, we have plenty of time to prepare, design, and write code. At first, the firewall was designed to use the received TCP packet to attack other teams which run the same service. It worked very well during the test environment, so we expected it would greatly help us in the actual game.

Part 3 - Project and PCTF

How did your team's project help or hinder your team's performance in the PCTF?

It turned out during the game; the service number was fixed. Those services' code and logic did not change; only the flag contained files that those services interacted with were added every tick. Because all services didn't

contain many lines, one of them was encrypted, and the other only had an executable. The team didn't use the vulnerabilities scanning script at all.

At the start of the game, due to more teams than in practice, there were too many requests and response packets for the firewall to be handled. Thus, it caused the firewall to be too busy. So as we stopped the firewall, an accident happened as all our services status was recorded as down. We tried to check the iptables rules, but all seemed fine. We tried to contact the instructor and ask for help. We think bugs may cause it in the VM or in our firewall, and we worried about that and did not open the firewall again. Things that happened later proved it was a wrong assumption, and our service was down for about five hours; we lost too much score due to both lack of defense functionality and services down. Five hours later, we tried to open the firewall again, and all the services were up and working normally. Then we realized that it was the delay effect of deleting iptables rules. All the PREROUTING traffic for our services kept being handled by our firewall even if we deleted the iptable rules. However, the rules will be effective immediately if a rule is added. Furthermore, we also analyzed the log file of our firewall program to look for the pattern of attack traffic. We found some attack patterns of other teams' solutions, who attacked our services. With this ability of firewall, we patched our services, copied their solution, and used it to attack other teams, even if we had not figured them out by ourselves.

Our team worked closely, analyzed the packet logs, and found the malicious strings which might appear in different service attacks. We monitored the service status, and our firewall kept running and dropping all packets with a malicious payload, which was filtered by a string array we got from different services, list as below:

- Backup service: [';cat', ';ls', '; cat', 'grep', 'cat ']
- Flashkids service: ['description%20like','description%20AS%20data', 'select%20description','union%20select']
- Simpleak service: ['1 aaaaaaaaaa', '/bin/','head --bytes','\x90\x90\x90]
- Saywhat service: (r'[0-9A-Za-z]{20}.json')

The auto exploit scripts also helped us significantly during the game. After we manually exploited the service successfully, we used the template and Linux's cron job to create the exploitation job automatically. Furthermore, we could focus on exploiting other services and not worry about the exploited service anymore since it would run every three minutes, find the flag, and submit it. It helped our team gain points stably on the un-patched services of others. Lastly, our rank rose significantly over time until the end of the game.

How could your team improve your project to perform differently in the PCTF?

We could do better on communication between teammates. It will make the team more efficient in cooperation with attack and defense. We often communicated via Zoom and Slack during the game, but each member participated individually online. So one member could not know what others were doing, thus leading to repeated tasks. There was a need for better coordination between all members, such as: reporting what one was doing, listing tasks based on services, who was doing which task, on offense or defense, etc.

Furthermore, we should spend more time on the test environment and try to exploit the sample service to get familiarized with the game. We spent a lot of time solving the problem with the firewall and the unavailability of our services. This led to a decrease in our time on analyzing and exploiting the services.

Part 4 - Peer Contributions, Collaboration, and Group Participation

How did each member on your team contribute to specific parts of the project and experience?

Although we divided our workload based on topics, each member researched individually and reported what he learned until the game. During the game, all members joined together to participate in the game.

- **Vo Minh Duy:**

  Duy is the captain of the team. He is responsible for preparing and submitting Project Proposal, Project Status Report, and Project Final Report. He kept track of the project's progress, informed other members about the report's required materials, and finally combined them into the final version. Duy also created the team's repository and organized all of the team's code.

  Duy focused on network and binary exploitation from the beginning of the project. Besides the course assignments, he was also doing Toddler's Bottle wargame on pwnable.kr and watching Professor Adam Doupé's walkthrough video of this wargame to gain more understanding. Duy wrote the flag submission script, which is a wrapper for swag-client, although the script was hardly used. Duy also provided the team with his researched knowledge, GitHub links, blog, and video tutorials about the CTF game.

  Duy participated in the PCTF the whole day. During the game, he hot fixed the team's backup service. When he was informed about the vulnerability in the backup service by a teammate, he located the vulnerability code and quickly inserted the check code, which is checking if the input contains follows word: “cat”, “ls” or “;” (Figure 3). This would prevent further exploitation from other teams. Duy also set up the cron job for auto exploitation of backup service run every three minutes.

```
void retrieve_backup()
{
   char cmd[200];
   char *retcat, *retls, *retbadchar;
   get_info();
   retcat = strstr(password, "cat");
   retls = strstr(password, "ls");
   retbadchar = strstr(password, ";");
   if (retcat!=NULL || retls!=NULL || retbadchar!=NULL){
        printf("Uh oh");
   }
   else{
        printf("Here is your backup data that was stored securely:\n");
        fflush(stdout);
        snprintf(cmd, 200, "cat %s_%s.secure.bak", name, password);
        system(cmd);
   }
}
```

Figure 3: Hotfix of backup service

In the analysis of the sampleak service, Duy provided the team with his exploit script. He had narrowed down the vulnerability of the binary application to its read_node function and computed the number of bytes the input needed for overwriting the function's return address. Duy provided information to the team, and together, other team members successfully exploited the service, built the script to automatically exploit other teams' sampleak service.

- **Kosuke Nagae:**

  Kosuke Nagae focused on application vulnerability, especially file access and command injection. Kosuke Nagae researched Metasploit and was the author of the vulnerabilities scanning script. At the beginning of the project, Kosuke Nagae provided the team with his idea of approaching the PCTF by dividing the workload based on topics. He also outlined the task needed for the project and set up Zoom for the team meeting.

  When the team discussed how to attack other teams and defend our service from others, Kosuke Nagae proposed that we duplicate the packets that other teams send to us to exploit our services and send the same ones after swapping our flag_ip to that of others. This idea was implemented in Jinyi Yu's script that was used as a firewall and duplicate attack.

  During the game, Kosuke Nagae tried to modify the teammate's script (exploit-sampleak.py) that sends packets to other teams and steals the flags from sampleak. Exploit-sampleak2.py consists of two parts, the first one is to put a file in the write mode before exploitation, and the second is to exploit in the read mode. The first part didn't work because the opponent always terminated the session, and the reason couldn't be found. This part was unnecessary because we needed to write a new file only once before exploitation in the write mode. The second part also was not completed because the address of the shellcode couldn't be found. Although his script did not work, his discussion about the service helped other teammates build another script that worked for the team.

- **Hongbo Wu:**

  Hongbo Wu focused on researching SQL injection, CSRF, and XSS attacks. He provided the team with a summary about the topic and the usage of SQLMap, which he used for SQL injection attacks.

  Hongbo Wu was responsible for exploiting and patching "flaskids" and "saywhat" web services under the web security category during the game. In addition, Hongbo Wu wrote "cronjobs" to run the exploit executable every tick to automate flag stealing and submission. To exploit "flaskids" service, the first thing he did after logging into the VM was to check source code and logs to look for web vulnerabilities. After looking into the source code of "flaskids" service, Hongbo Wu found potential vulnerabilities of SQL injections. Although the code had used some prepared statements to execute SQL, one of the queries to get "kids" was a concatenated string. And by checking logs, Hongbo Wu found in every tick, the admin would insert flag information into the "parties" table. In the end, he wrote an exploit script to send HTTP requests to inject "UNION" query to fetch data from "parties" table and stole flags. After successfully exploiting other teams' "flaskids" services, Hongbo Wu rewrote the query of getting "kids" by using prepared statements and patched this service. In this way, our "flaskids" service was secured.

  To exploit "saywhat" service, Hongbo Wu decoded the PHP file and found the command injection vulnerability of the code. The code allows users to include other files into the PHP file by sending the file path and name as the "page" parameter in the request to the server. So he quickly wrote a script to set the "page" parameter as "?page=../append/<flag_id>.json" in the URL to output all the flags in the response and stole flags. To fix this vulnerability, he updated the PHP file to sanitize and validate "page" parameters. In the end, he patched the service to secure our "saywhat" service from being attacked.

In summary, he has successfully exploited and patched "flaskids" and "saywhat" web services to steal many flags as well as keep our own flags from stealing by other teams. He also wrote some utilities and executables to facilitate flag stealing and submission.

- **Jinyi Yu:**

  Jinyi Yu focused on memory corruption exploitation and format-string vulnerabilities from the beginning. He practiced using pwn tools like ida, ghidra, gdb, and tried the previous pctf on the online source to prepare for the game. He also studied the previous ictf project source code in GitHub and the wiki knowledge of the pctf.

  Jinyi Yu was mainly responsible for the code writing of the firewall and part of scripts to crack the binary service. He was also provided the team with documentation about the firewall.

  Jinyi Yu communicated with team members to learn their ideas and ask for help and advice on the realization of our firewall. In the game, Jinyi Yu deployed the firewall, helped keep it running, monitored the running status and logs of the firewall, and solved the problem regarding the unavailability of all services. Furthermore, by monitoring the captured packet from our firewall, Jinyi Yu analyzed other teams' attack packets, reported to the team, and added more filter patterns to the firewall for further prevention.

- **He Jiang**

  He Jiang researched more profound on command injection (web vulnerability) and string-format vulnerability (binary vulnerability).

  During the game, He Jiang focused on exploiting vulnerabilities on backup and sampleak. Through the reversing analysis of two applications, He Jiang found some vulnerabilities (abuse of command system, buffer overflow, string format, etc.). After discussing with team members, He Jiang made a plan to exploit those vulnerabilities to capture the flag from other teams. In addition, he participated in the maintenance and patching of the server within the whole running time of the game.

  In Backup service exploitation, He Jiang analyzed the source code and found that the function system was a serious vulnerability in coding design (Figure 4).

```
void retrieve_backup()
{
   char cmd[200];
   get_info();

   printf("Here is your backup data that was stored securely:\n");
   fflush(stdout);
   snprintf(cmd, 200, "cat %s_%s.secure.bak", name, password);
   system(cmd);
}
```

Figure 4: Vulnerability in backup.c

Hackers could inject additional commands to bypass the original command and implement the read and write of sensitive documents. So, he tried to send some specific characters to read the file and successfully stole the correct flag from other teams' service.

In sampleak service exploitation, He Jiang also used Ghidra to decompile the file and find the vulnerabilities in the source code. Together with other teammates, he found suspicious code existed in function read_note. He exploited the buffer overflow attack in the input parameter. When he tried to execute the application, it leaked a random executable password every time. Analyzing that password, he could calculate all password addresses stored in the appended file. He injected shellcode at the specified address to implement the buffer overflow attack. With this, he successfully exploited the service and stole the flag information.