

# CAPTURE-THE-FLAG WRITE-UP

## PS-188/CS-50: CYBER SECURITY AND CYBER WARFARE

28 FEBRUARY 2017

ROMY ABOUDARHAM

YENNIE JUN

ABDI MOHAMUD

MACLYN SENEAR

### Executive Summary

---

Many common vulnerabilities continue to exist in various operating systems today. In this Capture the Flag challenge, we found several vulnerabilities in a Windows Server 2008 operating system – including common vulnerabilities such as susceptibility to SQL injection, susceptibility to cross-site scripting, weak password requirements, an open FTP port, and remote desktop being enabled – and we were able to first break into the system by accessing the remote desktop and guessing the weak administrator login credentials. Although this activity was performed on outdated software, we still learned about the consequences that could arise from not updating to newer versions with a higher level of security. Overall, it became apparent how important it is to track these vulnerabilities. For example, we saw how common vulnerabilities such as weak passwords are still very relevant today can have huge consequences when they fail to protect sensitive data. Our experience also shows that the simple step of practicing good cyber hygiene remains a critical component of preventing data breaches and unauthorized access.

### Introduction

---

Our team had the opportunity to hack into a system and find as many vulnerabilities associated with the system as possible. Our team included computer science majors who possessed previous experience with computer security, as well as international relations majors who possessed a background of cybersecurity policy but little technical background. We used our different skillsets to find and attack the vulnerabilities of the system given to us. We also learned from each other to increase our overall knowledge of computer security and policy.

### Tools and Methods Used

---

Our team used tools such as nmap, rdesktop, and whois in order to deduce vulnerabilities associated with the system. By using nmap, we were able to see what ports were open, and we used this information to detect vulnerabilities associated with the open ports. Using these tools, we were also able to figure out the operating system of the system as well. We further used methods such as SQL injections to break the system.

We divided the tasks according to the specific skill sets associated with each team member. Abdi,

who had the most technical skills associated with computer security, spearheaded the investigation to discover creative vulnerabilities. Maclyn, who possessed a broader background on security policy, was in charge of recommending various remediation and policy. Yennie and Romy investigated other tools to use to detect security vulnerabilities and ways to crack the system.

## Findings

---

### Operating System

The operating system in use was Microsoft Windows Server 2008 Standard.

### Services and Open Ports

The open ports and a brief detail of each port are listed in the chart below.

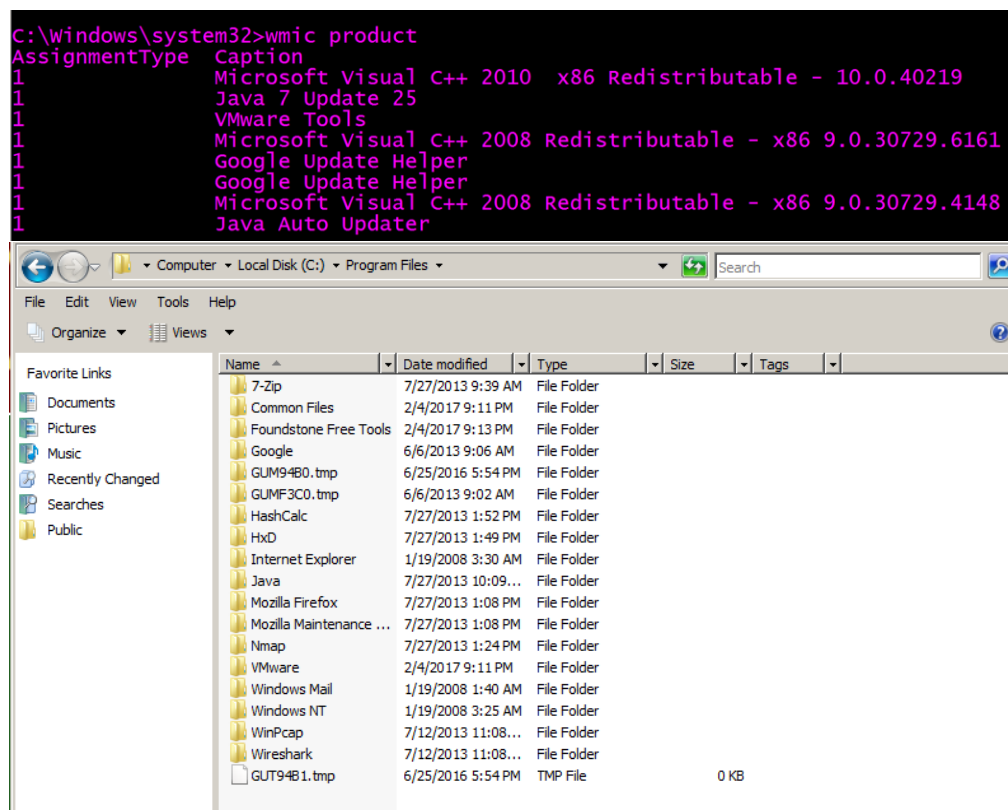
<i>PORT</i>	<i>SERVICE</i>	<i>DETAILS</i>
21	ftp	mapped by Windows Firewall
135	msrpc	Messenger service, exploited in popup net send messenger spam
139	netbios-ssn	NetBIOS Session service. TCP NetBIOS connections are made over this port
445	microsoft-ds	used for file sharing?
1025	NFS-or-IIS	Windows 2003/2008
1026	LSA-or-nterm	Nterm remote login network terminal
1027	IIS	Microsoft's IIS webserver
1028	unknown	
1029	ms-lsa	Local Security Authority (LSA) Authentication. Applications can use to authenticate and log users on the local system.
1030	iad1	(listening port for TCP)
3000	ppp	casino site
3389	ms-wbt-server	Remote Desktop
5357	wsdapi	Microsoft's Web Services on Devices API

## Username and Passwords Found on the System

- Due to port 3389 being open, we found that the remote desktop was enabled, which is a vulnerability discussed in a later section. This allowed us to gain access to the user login page where we discovered a very weak username and password combination:
  - username: Administrator
  - password: p@ssw0rd
- We knew that port 3000 was open, so once we had access to the remote desktop, we loaded the page "localhost/3000" which took us to a Casino website. We obtained all the data from the website's database called "users" by using .dump to create a snapshot of everything in that database. In this case, the information we found pertained to users:
  - (Note: Database: clean\_hacmecasino\_development.db, table name is "users")

```
sqlite> .dump users
BEGIN TRANSACTION;
create table users (login varchar(255), password varchar(255), chips in
rst_name varchar(255), last_name varchar(255), id integer primary key);
INSERT INTO users VALUES('andy_aces','2c4a1c243aca3390d5a8e87fef7c727d0
0','Andy','Aces',1);
INSERT INTO users VALUES('bobby_blackjack','65baec601446faf1481877a39e2
5465',10000,'Bobby','Blackjack',2);
INSERT INTO users VALUES('crystal_cardshark','d2ee721681569c7002d3fb094
3fc3d7',10000,'Crystal','Cardshark',3);
COMMIT;
```

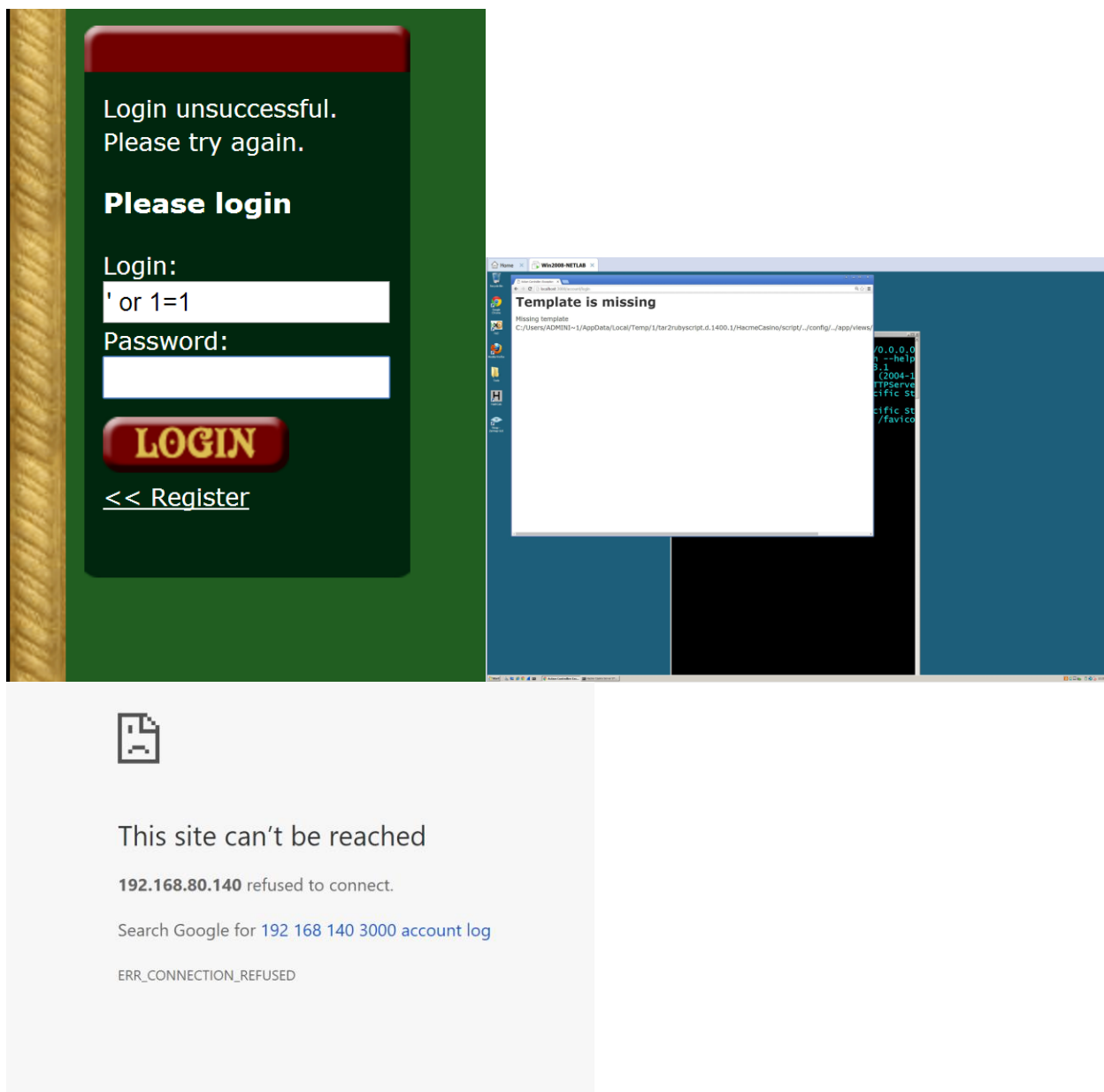
## Commercial Off-the-Shelf Software Packages

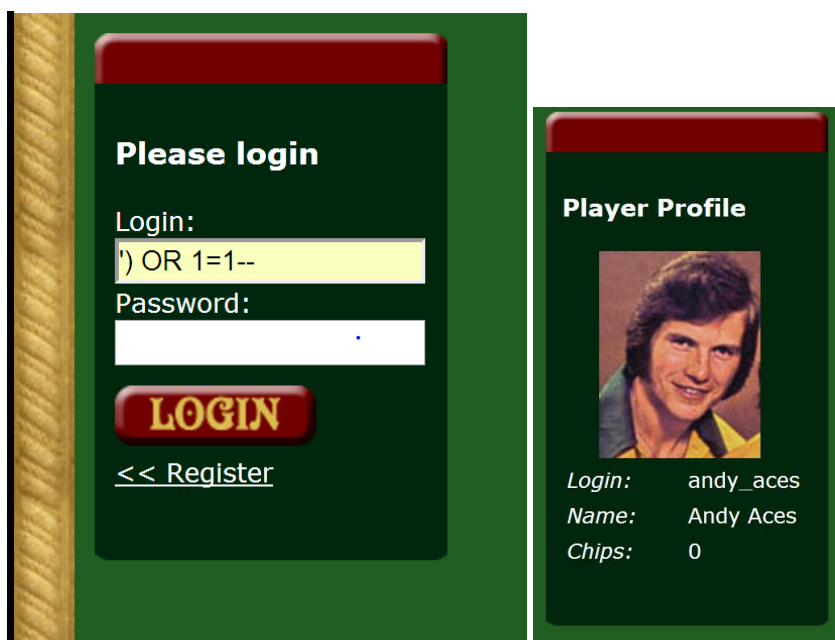


## Vulnerabilities

### 1. SQL Injection

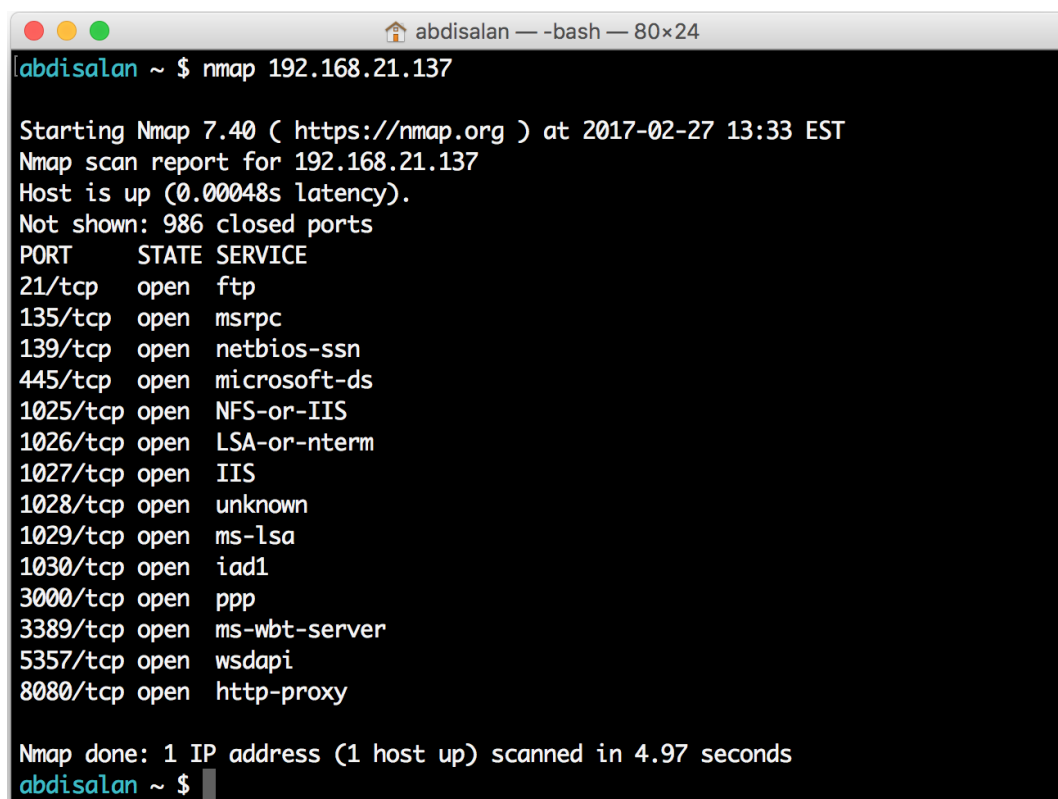
- a. *Description:* This is a common vulnerability in which the attacker can directly insert SQL commands and directly tamper with the original query. This can be dangerous because an attacker could potentially manipulate, delete, view, or add information stored in the database. We tried two different kinds of SQL injections. The first three photos demonstrate a blind SQL Injection, in which we crashed the site. The second three photos demonstrate a more targeted SQL injection, in which we can log-in maliciously as andy\_aces.
- b. *Screenshot:*





- c. *CWE/CVE IDS:* CWE-89 - Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
  - d. *The exact location of the vulnerability:* Log-in page of the site.
  - e. *The exploit or methodology used to find the vulnerability:* The exploit or methodology we used to find the vulnerability was a simple SQL injection - specifically, a blind SQL injection. SQL injections also caused the casino server to crash, which explains why it had to be restarted multiple times during the capture the flag class event.
  - f. *Recommendation to mitigate the vulnerability:* Sanitizing input data on the server side before executing the SQL queries.
2. **Weak Password(s)** (of the system)
  - a. *Description:* The password to break into the system as Administrator was a weak password known to have been used by John Podesta.
  - b. *Screenshot:* N/A
  - c. *CWE/CVE IDS:* CWE-521: Weak Password Requirements
  - d. *The exact location of the vulnerability:* Log-in page of the site.
  - e. *The exploit or methodology used to find the vulnerability:* The instructor gave us a hint that allowed us to figure out the Administrator password, but the weak password also made it susceptible to a brute force attack.
  - f. *Recommendation to mitigate the vulnerability:* (Paraphrasing CWE Potential Mitigations) Enforce a policy with specific requirements for user passwords, with criteria such as: minimum and maximum length; require mixed character sets (alpha, numeric, special, mixed case); do not contain user name; expiration, no password reuse; and (specific to this case) no commonly used passwords (such as any variation of "password") allowed.
3. **Open FTP Port**
  - a. *Description:* Intermittently, the FTP port would be open. This makes the server vulnerable to FTP Bounce Attacks. This is a vulnerability in which the attacker can map and port scan networks visible to FTP server.

b. *Screenshot:*



```

abdisalan ~ $ nmap 192.168.21.137

Starting Nmap 7.40 ( https://nmap.org ) at 2017-02-27 13:33 EST
Nmap scan report for 192.168.21.137
Host is up (0.00048s latency).
Not shown: 986 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1027/tcp  open  IIS
1028/tcp  open  unknown
1029/tcp  open  ms-lsa
1030/tcp  open  iad1
3000/tcp  open  ppp
3389/tcp  open  ms-wbt-server
5357/tcp  open  wsdapi
8080/tcp  open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 4.97 seconds
abdisalan ~ $

```

- c. *CWE/CVE IDS: CVE-1999-0017*
- d. *The exact location of the vulnerability:* The open FTP Port.
- e. *The exploit or methodology used to find the vulnerability:* With nmap, we can easily find out what ports are open. Port 21 would occasionally open, intimating that the FTP port was occasionally open.
- f. *Recommendation to mitigate the vulnerability:* Close the FTP port, since it is not being used constantly.

#### 4. Remote Desktop Enabled

- a. *Description:* This vulnerability gives anyone access to the remote desktop over the internet. This can be risky as attackers can guess login combinations through various methods. In our case, we obtained access to the remote desktop which served a page that wasn't live for outsiders to see. This enabled us to guess the username and logins and find out information about the site's users.

b. Screenshot:

```

abdisalan ~ $ nmap 192.168.21.137

Starting Nmap 7.40 ( https://nmap.org ) at 2017-02-27 13:33 EST
Nmap scan report for 192.168.21.137
Host is up (0.00048s latency).
Not shown: 986 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
135/tcp   open  msrcp
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1027/tcp  open  IIS
1028/tcp  open  unknown
1029/tcp  open  ms-lsa
1030/tcp  open  iad1
3000/tcp  open  ppp
3389/tcp  open  ms-wbt-server
5357/tcp  open  wsdapi
8080/tcp  open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 4.97 seconds
abdisalan ~ $

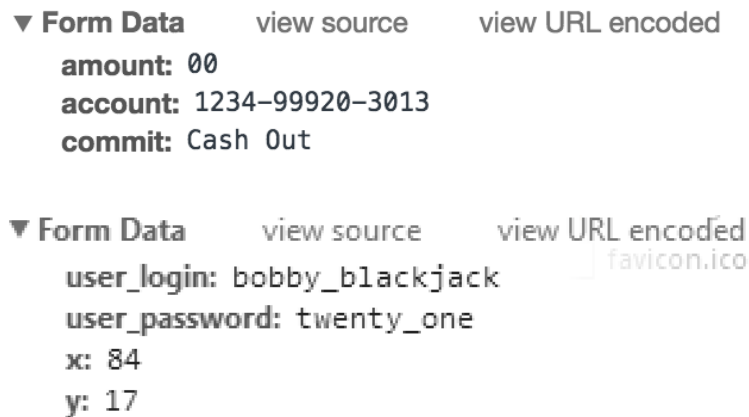
```

- c. *CWE/CVE IDS*: CVE-2016-0036: Remote Desktop Protocol (RDP) Elevation of Privilege Vulnerability
- d. *The exact location of the vulnerability*: The open Remote Desktop Protocol port
- e. *The exploit or methodology used to find the vulnerability*: We used nmap to view which ports were open. We knew Port 3389 was open, and thus knew that the port for Remote Desktop was open.
- f. *Recommendation to mitigate the vulnerability*: Remote Desktop service should generally be disabled to completely mitigate this vulnerability. If a user does need to use RDP, however, the user should take one of the following steps to secure the RDP services:
  - Change the listening port. By default, RDP listens on port 3389, but if one changes the listening port then attackers scanning the network will not see port 3389 open and will have a more difficult time determining if RDP is enabled.
  - Network Level Authentication (NLA) can be enabled to provide an extra level of authentication before establishing a remote connection.

## 5. Not Secure

- a. *Description*: The casino site sends data insecurely over HTTP. This causes a lot of sensitive information to be sent without any protection from the user. The sensitive information is sent as username and passwords when signing up or logging in, and bank numbers. The screenshots are from looking at the network requests done from the chrome browser.

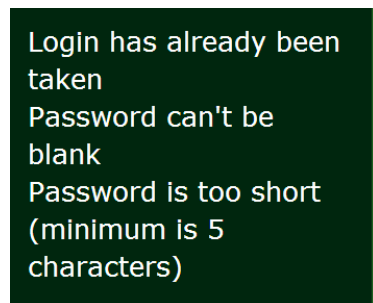
b. *Screenshot:*



- c. *CWE/CVE IDS:* CWE-319: Cleartext Transmission of Sensitive Information
- d. *The exact location of the vulnerability:* everywhere on the site, particularly when signing up, logging in, and cashing out.
- e. *The exploit or methodology used to find the vulnerability:* This exploit can be found by using any tool to listen to the network packets.
- f. *Recommendation to mitigate the vulnerability:* secure the site by adding HTTPS functionality to the site.

## 6. Weak User Passwords in Casino

- a. *Description:* The password requirements for users registering with accounts in the Hacme Casino are weak. For example, we made an account with the password 'password' and we were allowed to continue with the registration process without any roadblocks. The only requirements are that the password cannot be blank and must be at a minimum of five characters. Because the Casino takes weak passwords, it makes it easier for attackers to test for commonly-used passwords to hack into other user accounts.
- b. *Screenshot:*

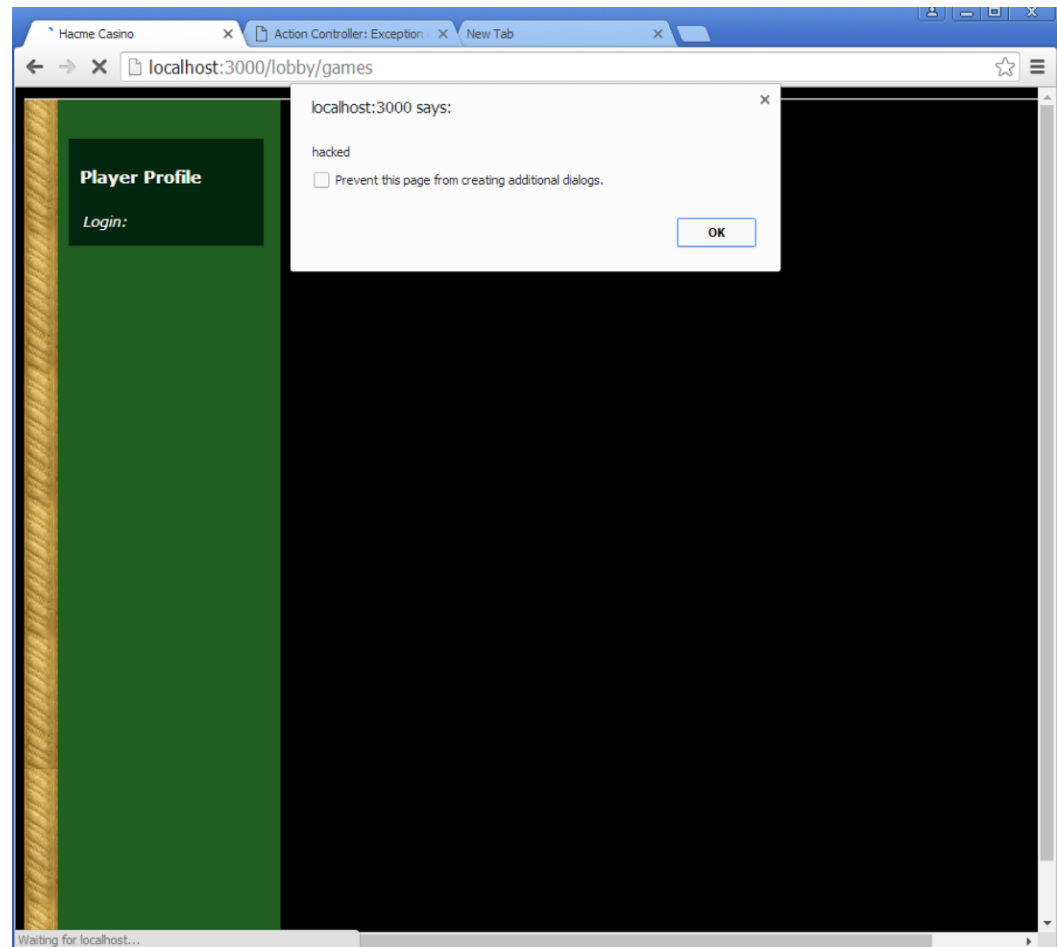


- c. *CWE/CVE IDS:* CWE-319: CWE-521: Weak Password Requirements
- d. *The exact location of the vulnerability:* Sign-up page on Hacme Casino (click "register" from the home page)
- e. *The exploit or methodology used to find the vulnerability:* Using a compromised password. This was found after getting access to the database, the passwords were stored as SHA1 hashes. We then used an online database of pre-generated hashes (hashkiller.co.uk) to search for the password. Only bobby\_blackjack's password was found.
- f. *Recommendation to mitigate the vulnerability:* Server can have more stringent password requirements, such as requiring special characters, case-sensitive values, and prohibiting commonly-used passwords.



## 7. Cross-Site Scripting

- a. *Description:* The user is able to add JavaScript to the registration form, allowing for arbitrary JavaScript execution where ever the username is presented.
- b. *Screenshot:*



- c. *CWE/CVE IDS:* CWE-79: Improper Neutralization of Input During Web Page Generation.
- d. *The exact location of the vulnerability:* This vulnerability is located on the registration page.
- e. *The exploit or methodology used to find the vulnerability:* To perform this exploit, create a script tag in the username field of the registration form. The JavaScript will be executed anywhere the username is displayed.
- f. *Recommendation to mitigate the vulnerability:* To correct this, the server must sanitize any and all user input.

## 8. Cross-Site-Request-Forgery

- a. *Description:* The casino site does not employ a system of verifying if a request came from the user or a malicious third party. In critical requests, such as transferring chips, the site does not require a Cross Site Request Forgery token to authenticate a request, only the session id. The result is that third parties can construct malicious commands, like a chips transfer and trick the user into executing these commands unintentionally.

The example in the screenshot is a URL, where if anyone with an account on the Hacme casino uses this link then they will send money to Andy Aces.

- b. *Screenshot:* Notice in the URL that there is a constructed link that sends 20 chips to Andy

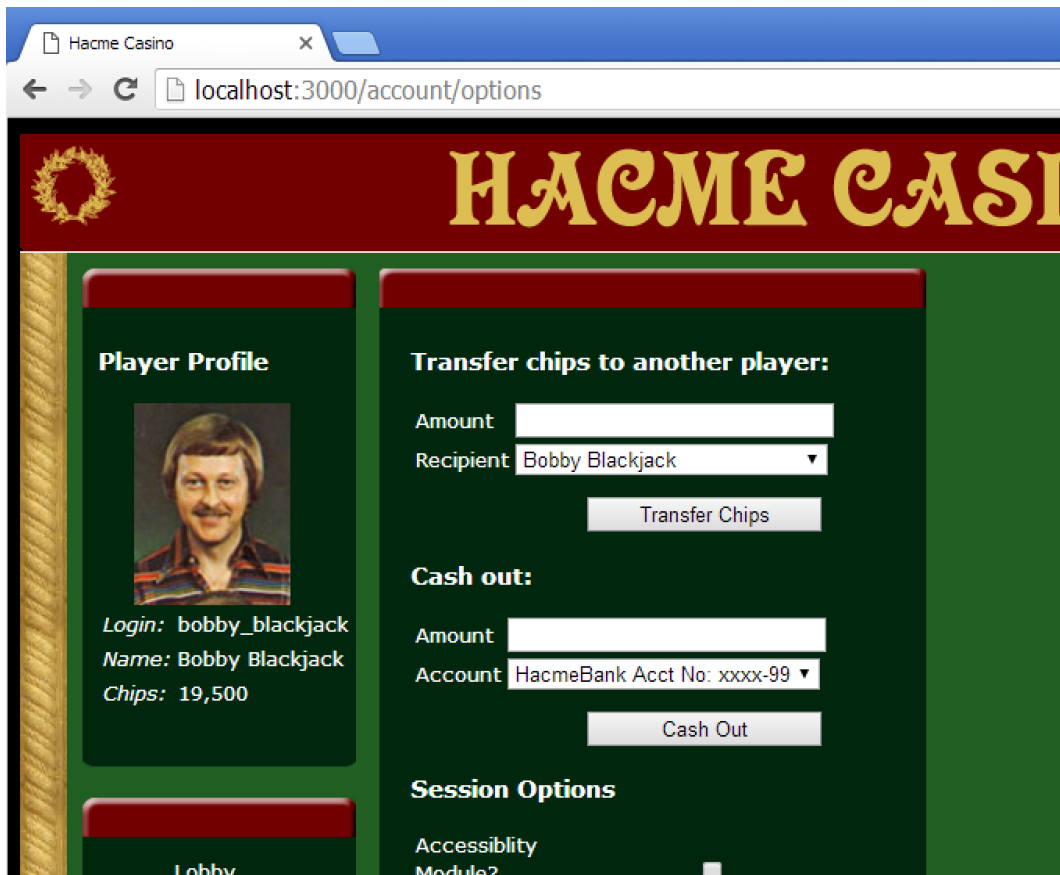


- c. *CWE/CVE IDS:* CWE-352: Cross-Site Request Forgery
- d. *The exact location of the vulnerability:* This can be done anywhere as long as the victim clicks the link.
- e. *The exploit or methodology used to find the vulnerability:* The exploit is performed by creating a URL with the endpoint `/account/transfer_chips` and adding the parameters `transfer` for the amount to transfer, `login[]` for the username to send to, and `commit` which is set to "Transfer Chips"
- f. *Recommendation to mitigate the vulnerability:* A solution to this vulnerability is to add a system of adding csrf (cross site request forgery) token to all forms where the client can confirm if the form sent to the server was generated by the site and not a malicious third party.

## 9. Allows user to send themselves chips

- a. *Description:* The transfer chips form can be changed to make the sender and receiver of chips be the same user. The effect is that on the first attempt, the user is given double the chips they send to themselves and all subsequent attempts lose any chips they send to themselves.

- b. *Screenshot:* Notice that the user and the recipient have been edited to be the same.



- c. *CWE/CVE IDS:* CWE-345: Insufficient Verification of Data Authenticity
- d. *The exact location of the vulnerability:* The location of the exploit is on the options page in the transfer chips form.
- e. *The exploit or methodology used to find the vulnerability:* The exploit is performed by editing the HTML form and writing your username as one of the possible recipients. Once you add your username, you can send yourself chips.
- f. *Recommendation to mitigate the vulnerability:* A recommendation is to fix this is to add a subroutine on the server side to check if the sender and recipient are the same users and not allow any requests where they are the same.

## Questions

As we worked on our project, our team thought of a few questions.

1. What are some of the other easily exploitable vulnerabilities on this system that we were unable to find?
2. The CTF we had didn't require us have a deep understanding of Metasploit, nor require us to use software such as Metasploit. What more deeply technical vulnerabilities could we have found had we used a software such as Metasploit?

## Connections

The vulnerabilities that were uncovered were vulnerabilities commonly found in other systems that have previously been hacked. For instance, SQL injections and weak passwords are common vulnerabilities. For more details regarding real-world vulnerabilities and responses, please see the “Remediation and Policy Description” section below.

## Reflections

If we had more time, we would have continued to look for more vulnerabilities in the system. Specifically, tried to find and execute any exploits on the FTP port that was intermittently open.

Also, we would have not tried to use Metasploit from the beginning and tried common attack methods first like guessing weak passwords, SQL Injection etc. We lost a lot of time trying to get Metasploit to work and it ended up failing to produce any results.

## Remediation and Policy Discussion

---

Several of the vulnerabilities we found on this Windows Server 2008 system are unsurprisingly some of the most commonly exploited weaknesses in the world. Despite SQL injection being named the No. 1 most critical vulnerability by both the Open Web Application Security Project (OWASP) in its 2010 and 2013 reports and the 2011 SANS/CWE 25 Most Dangerous Software Errors report, SQL injection remains one of the most commonly used exploits by attackers and has figured prominently in recent large-scale data breaches such as the Panama Paper scandal.<sup>1</sup> In this incident, unidentified attackers breached the systems used by Panamanian corporate services firm Mossack Fonseca in 2015 and released documents that chronicle how wealthy individuals around the world used offshore companies to conceal their ownership of assets and avoid paying taxes to their respective governments, and this data breach has officially been classified as the largest in history. Post-incident analysis revealed that the content management system used by Mossack Fonseca’s Client Information Portal contained at least 25 vulnerabilities, the most prominent among these being a susceptibility to SQL injection, which investigators speculate may have been one of the techniques used by the attacker(s). Additionally, this Client Information Portal had last been updated in 2013, and the version of Microsoft Outlook Web Access used by the firm had last been updated in 2009, which is largely why most of the documents released came from its unencrypted email traffic.<sup>2</sup>

---

<sup>1</sup> Paul Ionescu, “Inside the Mind of a Hacker: Attacking Databases With SQL Injection,” *Security Intelligence*, May 12, 2016, <https://securityintelligence.com/inside-the-mind-of-a-hacker-attacking-databases-with-sql-injection/>

<sup>2</sup> Neil Jones, “Lessons Learned From the Panama Papers: 10 Convenient Ways to Improve Your Security Protection,” *Security Intelligence*, May 2, 2016, <https://securityintelligence.com/lessons-learned-from-the-panama-papers-10-convenient-ways-to-improve-your-security-protection/>

Such a vulnerability should be considered especially egregious given that a SQL injection-based attack was also used in the 2008 Heartland Payment Systems data breach, in which cyber criminals stole information from 100 million credit and debit cards and which at the time was the largest breach in history.<sup>3</sup> In the case of the server used for this CTF game, we were unable to penetrate the system using an SQL injection, but the fact that a simple SQL injection used on the site's log-in page crashed the site reflects how this well-known vulnerability continues to plague sites containing secure data. Our recommendation for mitigating this vulnerability--sanitizing user input data on the server side before executing the SQL queries--is a relatively simple solution that has been recommended for over a decade, yet corporate sites still fall victim to SQL injection attacks almost routinely, suggesting that private and public organizations can do a great deal to protect themselves from data breaches, unauthorized access, and site crashes merely by focusing on preventing relatively unsophisticated and not very innovative attack vectors.

This same principle is supported by some of our other findings from the CTF exercise, such as the fact that we easily gained access to an administrator account for the system through remote desktop using the relatively weak username and password combination of "Administrator" (which is the default administrator username, easily found through Google) and "p@ssw0rd." Though such weak credentials may seem farcical and defy the simplest tenets of good cyber hygiene, this vulnerability mirrors an unfortunately highly prevalent issue among users of secure systems. A study done by Praetorian between 2013 and 2016 found that researchers were able to compromise client systems 66% of the time using weak domain user passwords.<sup>4</sup> Investigators believe that the 2015 attack on the U.S. government's Office of Personnel Management (OPM) involved the use of stolen credentials from a government contractor, allowing Chinese cyber thieves to steal the personnel files of more than 20 million government employees and others who had undergone background checks. The post-mortem on this incident revealed that many OPM employees were using weak usernames and passwords, which may have compounded the breach.<sup>5</sup> Additionally, secure databases within the system, such as those containing government employees' social security numbers, were not encrypted partly because, as OPM Director Katherine Archuleta acknowledged, many of the legacy networks were too old to support it. A lack of encryption within the system's casino site, which sent information using HTTP, allowed us to steal sensitive information such as the usernames, passwords, and bank account numbers of other users. In the OPM case, encryption may not have done much to mitigate the theft of data since the attackers gained access using legitimate credentials, but encryption of the data still should have been one of the primary security measures taken. After the incident, OPM modernized its systems and required employees to use stronger passwords and to log in using two-factor authentication, which are similar to our recommendations to mitigate the vulnerabilities we found on this system.

We would not have been able to exploit the weak administrator username and password vulnerability, however, had remote desktop not been enabled for the system. As Kaspersky Lab

---

<sup>3</sup> Kevin Judge, "Heartland: The Greatest Network Breach in the 21st Century," *Toolbox.com*, October 24, 2012, <http://it.toolbox.com/blogs/internet-security/heartland-the-greatest-network-breach-in-the-21s-century-53445>

<sup>4</sup> Josh Abraham, "How to Dramatically Improve Corporate IT Security without Spending Millions," *Praetorian*, <https://www.praetorian.com/downloads/report/How%20to%20Dramatically%20Improve%20Corporate%20IT%20Security%20Without%20Spending%20Millions%20-%20Praetorian.pdf>

<sup>5</sup> Brian Naylor, "One Year After OPM Data Breach, What Has the Government Learned?" *NPR*, June 6, 2016, <http://www.npr.org/sections/alltechconsidered/2016/06/06/480968999/one-year-after-opm-data-breach-what-has-the-government-learned>

reported in June 2014, RDP attacks using brute force are on the rise, largely because many users still employ short, generic passwords that can be guessed through repeated trial and error.<sup>6</sup> Our successful penetration of the system in the CTF exercise, exploiting the enabled RDP connection vulnerability which was compounded by the weak administrator username and password vulnerability, represents another one of the most commonly used attack vectors that is easily preventable.

In short, the system we targeted in this CTF exercise replicated many of the most commonly exploited vulnerabilities in the real world, and the experience highlighted how most common attacks are relatively unsophisticated and take advantage of easily prevented vulnerabilities, such as failures on the part of users and known vulnerabilities within outdated software or networks. The exercise also illustrated how the universal utilization of best cyber hygiene practices--such as requiring stronger usernames and passwords of users, ensuring that organizations update software as often as they should and install patches for known vulnerabilities (i.e. with regard to both the Panama Papers and OPM breaches), and ensuring that developers learn from the past mistakes of other developers (i.e. with regard to SQL injection vulnerabilities)--would go a long way in thwarting most cyber attacks. Before worrying about zero-day attacks or other sophisticated vectors, a good cyber policy for any private organization or government agency should first focus on addressing the basics.

## Conclusion

---

Overall, our team managed to successfully break into the system. Given only the I.P. address, we were able to discover the operating system of the machine, the open ports and services, and many common vulnerabilities such as susceptibility to SQL injection, cross site scripting, weak password requirements, an open FTP port, and remote desktop being enabled. The speed at which we discovered these vulnerabilities was especially notable because it became clear how quickly with ease an insecure system can be hacked into. Once we were in, we were able to extract data and harming the site by forcing it to crash. One of the big takeaways from this activity has been that some of the vulnerabilities we found on the Windows Server 2008 system are notably some of the most commonly exploited cyber-security vulnerabilities. We see these weaknesses present in today's current events where data breaches flood the news. It's clear that cyber security holds no guarantee that your information is safe on the internet. That being said, this isn't something to ignore or take lightly as we are becoming more and more dependent on the internet for storing important data. Recognizing the risks and ways to improve your own protection as best you can should continue to be emphasized and acted upon.

---

<sup>6</sup> Anton M. Ivanov, "A multiheaded battering ram: RDP Bruteforce attacks on the rise," *Kaspersky Lab*, June 27, 2014, <https://business.kaspersky.com/a-multiheaded-battering-ram-rdp-bruteforce-attacks-on-the-rise/2145/>