Security Analysis of Terra:Battle for the Outlands

Neeraj Prashar (neeraj.prashar@gmail.com) [1], Cloud Shao(cshao3@interchange.ubc.ca)[2], Adnan Jiwani(jiwaniadnan@gmail.com)[3] and Arash Malekzadeh(arash8m@gmail.com)[4]

[1]Software Engineering, University of British Columbia, Vancouver, BC V6T1Z4 Canada
[2]Computer Engineering, University of British Columbia, Vancouver, BC V6T1Z4 Canada
[3]Computer Engineering, University of British Columbia, Vancouver, BC V6T1Z4 Canada
[4]Software Engineering, University of British Columbia, Vancouver, BC V6T1Z4 Canada

***Abstract*—Tasked with performing a security analysis of the massively multi-player online game Terra: Battle for the Outlands, we completed a threat analysis on the product. User experience is the most significant asset at stake. Cheaters, hackers, and identity thieves all pose threats. As a result of our experimentation, we found that the system is vulnerable to several attacks including online dictionary attacks, password snooping, denial of service, API hooks, and unauthorized e-mail sending. The system was reasonably strong against SQL injection, packet injection, DLL injection, database-specific (MySQL) vulnerabilities, and race conditions. We discuss the principles of secure systems that were violated causing these vulnerabilities and provide insight on how to build more secure systems using these findings.**

***Index Terms*—Client-server systems, Computer Crime, Games, Security**

I. Introduction

Online games are a class of software that is purely for entertainment, yet heavily dependent on security. When attackers are able to break the security and cheat within a game, the entertainment value of the game can drop significantly. This makes the experience far less enjoyable for users and effectively undermines the software's purpose. In addition, identity theft in any application in the online world can result in the theft of the same user's identity in other applications.

A. Significance

The significance and impact of our research is not limited to this particular online game. Developers often have similar habits when writing software, and even if they do not, security holes could lie in third-party libraries that the developers have little control of. Games are a $6 billion market that grows at 10% per year and according to Microsoft, gaming is the third most common activity on its platforms [1]. As such, the stakes here are higher than one might expect.

Massively multi-player online games, with their huge numbers of concurrent and overall users, are on the cutting edge of distributed software. These are some of the first large-scale distributed real time systems in the world, and what better proving ground is there for such new technology than a game? Developers should benefit from paying attention to the developments happening here before they soon become wide spread in other areas of software. As Hoglund and McGraw put it, "the security story in the gaming world may be a harbinger of things to come in the rest of the online world" [1].

B. Background

Terra: Battle for the Outlands was developed originally by Kaon Interactive, and most recently by TerraOutlands. In the game, players gain experience points, claim land, and build their own assets. All other players exist in the same persistent game world where their assets exist and need to be defended even when the players are not logged in. Currently the game is free with an optional donation.

Signup is done through an online form. Players enter their email address and are sent an account creation key, which they use to create a new account. Players then download the game installation executable and install the game client to their hard drive. To start the game, a player enters his/her credentials into the game client. The client application is only available for Windows x86.

C. Assets and Threats

For developers of the game, there is often significant monetary value in the successful operation and intellectual property of the game. For users, it is as important to preserve the integrity of their identity and the secrecy of their credentials as it is in many other online applications. The primary assets at risk are the user experience, income for the game studio, and the user's online identity.

User experience is not a tangible asset that is represented by any specific resource or function. It is difficult to assign a specific value to this asset and the best we can do is say that the game studio's income from the game (on the order of $1000's) is at stake if users are unhappy. We expect the primary threat agent in most of these cases to be malicious users intent on cheating.

All CIA (Confidentiality, Integrity, Availability) properties are covered by these threats. Confidentiality is threatened by snooping and spoofing. Modification of game state threatens integrity and availability is threatened by the possibility of DoS (Denial of Service).

Users' identities in the game are not necessarily of significant value. User credentials, however, have been shown to be consistent across different systems. According to "A large-scale study of web password habits" by Florencio, D. and Herly, C., the average password is shared across 3.9

different sites [2]. This means that a compromised identity on Terra can lead to compromise of the same user's identity elsewhere. Threat agents for this threat include identity thieves and malicious users.

II. Analysis

In this section we will describe the Terra system, its composition, and its security properties from an outsider's perspective. We listed all the attacks that the system could hypothetically be vulnerable to during the proposal phase of the project, and here we will provide that list. We will then attempt to provide insight on which attacks worked (or didn't work) and why the vulnerabilities existed (or didn't exist).

A. Description

Terra's backend comprises of only one web server running Apache 2.2.9, on a Linux (Fedora) based operating system with ports 22 (ssh), 25 (mailserver), 80 (http), 541 (osiris), 3306 (mySQL), 8801 (Firewall). Its alias name is terracorps.com and the actual hostname is terraoutlands.com. The authoritative domain name servers that have a Resource Record (RR) of terra are ns51.domaincontrol.com and ns52.domaincontrol.com. This information was obtained using primitive Unix tools like nmap and dig.

B. Proposed Attacks

There were a number of attacks attempted on both the TerraCorps website and the game. The following subsections list and explain each type of attack.

1) SQL Injection

SQL injections are one of the major threats to any system that stores information. According to Art Wittmann's article "The Fastest-Growing Security Threat", "the number of SQL injection attempts has gone from a few thousand a day last year to more than half a million a day now" [4]. In SQL injection, the attacker inserts SQL statements as input to the application which are later processed by its database management system. If the application does not validate inputs, the attacker's SQL statements can execute and produce results that an attacker can analyze to determine the database structure of the application. He/she can then write specific SQL statements that retrieve critical information stored on the attacked system.

2) Password Brute Force

"An ideal password is something that you know, something that a computer can verify that you know, and something nobody else can guess – even with access to unlimited computing resources" [5]. In practice, humans typically cannot remember more than 8 character passwords with uppercase and lowercase letters, numbers, and special characters such as "!@#$". As a result, many passwords can be cracked by trying different combinations of characters mentioned above. This strategy of using computing resources to crack a password by trying all different combinations is called brute forcing.

3) Packet Injection

Packet injection exploits communications between the client and the server. The client sends the server a packet requesting certain information. The server, upon receiving the packet, replies back to the client with the information requested. In Packet injection the attacker simply creates a packet on behalf of the client, without client's knowledge, and passes this packet to the server. The server, oblivious to who created the packet, replies back with information that the attacker requested.

4) Denial of Service

DoS (Denial of Service) is a form of an attack that prevents one or many users from accessing the resources and services of a server. This attack is of low technical impact and does not threaten data integrity and confidentiality, but it can negatively affect the company. Users would be less inclined to donate funds when the service is not regularly available. Reduced donations mean less development and advertising money for the company.

Denial of service attack can be done on either the server or the client. In the case of server DoS, the attacker floods the server (online system) with numerous false requests to overload it. The server becomes unavailable to actual legitimate clients that want to use the online system. In the case of client DoS, the attacker gains access to the client's account and prevents it from connecting to the server.

5) Memory Analysis

Being an online multi-player game, competition could promote cheating in the game. Using Memory analysis, one can manipulate the game's memory such that it gives one an advantage over his/her opponents. For example, the attacker can modify the memory address which stores an ammunition value so it never decreases even when the attacker fires in the game.

6) Password Snooping

In a snooping attack, the attacker simply inspects packets going between the client and the server to discern some useful information. In terms of password snooping, the attacker specifically looks for the packet in which the client sends its login credentials to the server for verification. Once the attacker has the login details he/she can later impersonate the client and have access to the online resource.

7) E-mail Impersonation

In email impersonation the attacker first remotely connects to a company's mail server and then using mail protocols such as SMTP sends an email to the employees at the company pretending to be a co-worker. Since the emails come from the local servers, the company's defense mechanisms (anti-virus programs, filter, etc) are more likely to let the email go even if it contains malicious content. Once the employees get the email, they too trust internal mail and are more likely to either execute the malicious content in the mail or even reply with requested information in the e-mail..

8) DLL Injection

Dynamic Linked Libraries (DLLs) are libraries that contain executable instructions that can be used by windows applications. In DLL injection the attacker first figures out

which DLLs the system is using during execution and then removes that DLL and loads a malicious DLL that if executed would exploit or crash the system.

9) Race Condition

Race conditions occur when "a transaction is carried out in two or more stages, and it is possible for someone to alter it after the stage that involves access rights" [6]. This type of attack seeks to exploit a lack of complete mediation. An attacker exploits such transactions to gain access to protected resources in the system memory, to either break down the system or get an upper hand if the system is a game.

10) Buffer Overflow

In this attack the attacker sends a string of size for example 2N as an input to a system. The string is then sent to the server that copies it to a buffer that has a length of N characters. Since 2N is greater than N a buffer overflow occurs. Buffer overflows can be used in many different ways depending on how the server deals with them. For example, if the server simply crashes on a buffer overflow then this can be used to launch a DOS on the system.

C. Vulnerabilities

The following subsections detail the vulnerabilities we found in both the game and the web server, the tools we used to find them, and the methodologies employed.

1) Denial of Service

Two different DoS attacks were performed on the system. Neither was an "attack" per-se; they were only side effects of defenses against our other attacks. The first one was caused by Terra putting an IP range ban on the entire UBC campus. This was performing result of us trying a password brute attack, which created 60 HTTP requests (TCP connections) at a time. The attack ran for seven hours and we believe that administrators at Terra may have detected it and banned our IP range, inadvertently cutting off access to all users at UBC.

The second attempt to cause DoS involves individual user accounts. A player can be suspended from playing the game if they use a known cheating tool. Using an experimental account we created, we played the game using victim's username and password, and also used a program called "Cheat Engine" which resulted in the user account getting suspended. By using another player's credentials, we can get them banned from the system in this way.

2) Password Brute Forcing

Terra's website enforces the rule that users must choose user names and passwords with a maximum length of 8. As a result, the chosen passwords tend to be a short combination of English words and/or digits. Since TerraCorps allows users to have an unlimited number of tries for guessing the correct password in order to login, the website is vulnerable to password brute forcing. Using a brute forcing tool called "Brutus" it is possible to perform a dictionary attack on one or many users. Although this attack is slow due to the fact that it deals with HTTP requests, we were able to find the password of 15 users out of a list of 100 users. The list was retrieved from TerraCorps website by accessing the clan member page. The attack took about 12 hours to complete and was run on one computer at a time. In an effort to avoid setting off automatic firewall alarms, the maximum number of HTTP requests was set to 10.

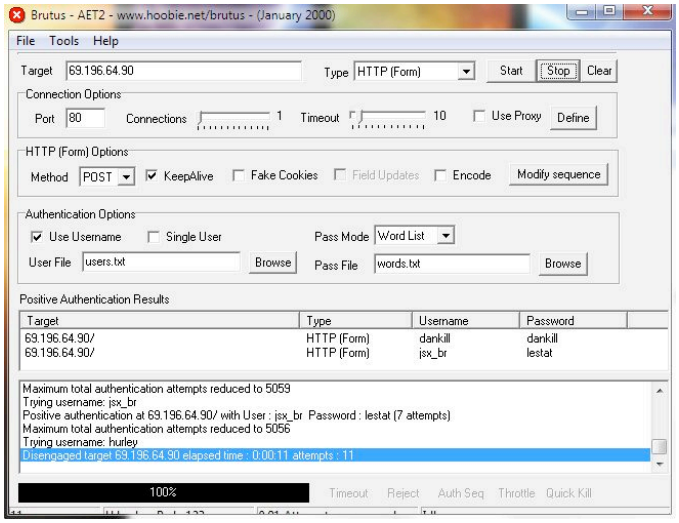

FIG. 1 Screen shot of Brutus

3) Password Snooping

Although this attack (in its experimental form) requires the victim and the hacker to be under the same network subnet, it is still of great importance. When logging in to the website, usernames and the passwords are sent to TerraCorps server in plain text. This means that anyone with the ability to sniff HTTP request packets can view the username and the password in those packets.

In order to monitor the packets being sent out from a particular computer, we took advantage of a technique called ARP poisoning to perform a Man In the Middle Attack. ARP is a protocol used within subnets to allow computers identify themselves to the router. For example one computer would ask "who has IP x.x.x.x". The computer with that IP would respond by sending its MAC address. "Cain & Able" is a tool that offers ARP poisoning. Using this tool, it was possible to tell the victim's computer that we are the router, and tell the router that we are the victim.

After establishing ourselves as the Man In the Middle, we were able to see the HTTP packets coming from the victim's computer, which contained the username and the password. This can also be achieved using a WiFi card that is capable of sniffing packets in promiscuous mode. The figure below displays the player's username and password in a packet, which is "dankill" in this case.

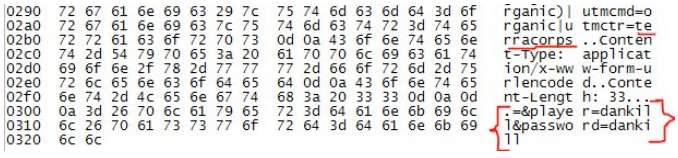

FIG. 2 Snooping Usernames and Passwords

4) Telnet/SMTP

It was discovered that the TerraCorps server has port 25 open, which is a known port for telnet access. Using the system's telnet server we were able to connect to their mail server and send emails internally to any employee with the email address of xxx@terraoutlands.com. The mail server only allows sending emails to the addresses ending with "terraoutlands.com".

There are two threats associated with this vulnerability. One threat is that it is possible to send fake emails containing malicious URL's to the employees from a trusted source such as admin@terraoutlands.com. As it is shown in Fig.3, since this message is being sent from the TerraCorps mail server, it would be far less likely to be marked as junk. Though the use of those e-mails to perform further attacks is not within the scope of our project, those e-mails sent could contain some malicious content, which could result in exposure of the company's private data.

This brings up another vulnerability in that it allows hackers to verify if a particular username@terraoutlands.com exists. This vulnerability enables us to obtain a list of users and possibly perform a brute force attack to hack accounts that belong to employees who could have higher privileges.

FIG. 3 Unauthorized Mailserver Access

5) API Hook (Speed Hack)

Speed hacks exploit the fact that the game client uses calls to the Windows API for timing purposes. Those calls can be intercepted, and the result can be multiplied by some factor. This multiplies the amount of time passed by some factor so that the game client believes that more time has passed, resulting in the game progressing faster. There are many implementations of speed hack available on the internet. We used the implementation integrated into a multi-purpose cheating tool called "Cheat Engine" which gave us the ability to fire faster and get through some in-game sequences faster such as running to our vehicle. From our observations, we believe that Terra does not have any automatic mechanism for detecting speed hack, though it does have the ability to detect some other side-effects of Cheat Engine.

D. Strengths

This section outlines all the unsuccessful attacks that were launched against the web server that hosts Terra, and briefly describes the techniques and tools used, as well as the results obtained and how it reflects the security level of the web server.

Like any online, centralized-server based game that has a large user base, there is bound to be some sort of a backend database that records user information, account status, player records, etc. This opens up the possibility to exploit the database server by trying SQL injection attacks. These attacks try to exploit unverified, malicious user-crafted SQL statements that attempt to run SQL code against the web server.

Using nmap4, we found that port number 3306 was open for database queries and was running version 5.077 of MySQL database server. Using Perl, a simple script was written to connect to the database, and attempts were made to match dictionary passwords with usernames like admin, root, user, etc. As a response to the initial challenge request, a hand-crafted packet with bogus data was sent. However, the web-server was able to handle these packets, most of which were on average 6000 bytes, and still returned could not connect message.

In the "Lost password" section on Terra's website, a user can request to be sent his account information if he forgot that. Several crafted replies like

'some-text' OR 'x' = 'x

were tried in order to trick the web server to send a user's information to any e-mail address. It turned out that the web server does not take dynamic information such as the e-mail and username using the HTTP Get and Post Methods and instead parses everything using a Perl file (account_management.pl) and then sends this information in some sort of a packet to be authenticated with the backend server. An open source tool called sqlmap was used to try more sophisticated SQL injection attacks.

Terra was protected from DLL attacks in the sense that the terrain maps and user information was loaded when one logged on using tcp/ip packets. Contrary to our assumption, the game was not using operating system DLLs while in use, even though the actual game did use a few DLLs for installation. Due to this, we could not perform our attack to modify OS drivers to see through mountains or avoid fireballs or become invisible when being fired on [3].

Handcrafted TCP/IP packets along with their headers were also sent to the server simultaneously while having a game session open to see if we can disrupt user experience either by lagging the game or somehow break an existing connection. We were greeted with a "bad handshake" message as a receipt of the packet.

Buffer overflow attacks were attempted manually against all possible user input fields, however, it seems like all the text fields only allowed up to a certain number of characters to be entered and almost every user input field was being parsed through a Perl file instead of using manual GET and POST methods. The attacks were attempted using code written in java, which used reply and request objects to send the data to the web server.

III. Discussion

In this section, we will briefly list the basic principles of designing secure software systems. Principles that we felt

were violated will be discussed in detail and suggestions will be provided on building systems which might better fulfill those design principles.

A. The Principles of Designing Secure Systems

The principles of designing secure software are guidelines for building any type of online system. They are not absolute rules and not all of them can be applied to every system.

For convenience, listed here are the ten principles of designing secure software (from the EECE 412 Principles of Designing Secure Systems Lecture Slides):

1. Least Privilege
2. Fail-Safe Defaults
3. Economy of Mechanism
4. Complete Mediation
5. Open Design
6. Separation of Privilege
7. Least Common Mechanism
8. Psychological Acceptability
9. Defense in Depth
10. Question Assumptions

In the following paragraphs we will discuss the principles of designing secure software that we felt may have been violated, along with countermeasures. The reader must keep in mind that as outsiders of Terra, some of our arguments may represent only our best guess about the system given the evidence we see.

B. E-mail Impersonation and Countermeasures

On an open interface like telnet the principle of **least privilege** should be followed. Anonymous users should possess the least privilege necessary, which in this case is likely no privilege at all. If the interface was actually used for some purpose, it would make sense to create other users on the system and require authentication for them to perform their tasks. This is also related to **separation of duty**; if an anonymous user was able to log in to the system, it would make sense to require separate credentials in order to send e-mail.

C. Online Brute-Force/Dictionary Attack and Countermeasures

The system, through the sign-up process, limits user names and passwords to eight characters. **Economy of mechanism** states that the system and its mechanisms should be as simple as possible. It would be beneficial to question the purpose of the limit and evaluate whether the system would be more or less complex as a result of removing it. **Psychological acceptability** also suffers when users are limited to eight characters. Allowing users to create longer passwords and encouraging them to use more secure passwords containing different types of characters would be the best countermeasure.

The other problem is that continuous retries are allowed. The system **incorrectly assumes** that if a user tries to log in and fails, then they must have simply mistyped the password. We recommend that exponential back-off or jailing be used to prevent continuous retries.

D. Password Snooping and Countermeasures

When dealing with snooping, the principle of **least common mechanism** is often brought into question. In this case, HTTP was used to transfer login credentials. The **assumption** might have been that the channel could be trusted, which was incorrect as illustrated by our fairly simple ARP poisoning demonstration. The use of a secure channel such as SSL for authentication would have prevented this problem. Users can partially protect themselves by connecting with internet routers through secure channels only.

E. API Hook (Speed Hack) and Countermeasures

In order to do timing, the game uses Windows API calls. In our opinion, the bigger issue is that the game client cannot be **assumed** to be trustworthy since it can be controlled by an attacker. We found that the player position seems to be verified on the server side, but not every action is verified by the server. For example, the client can rotate its turret or fire its gun as fast as it likes without being pulled back by the server. Going one step further, we noticed that when using the speed hack, our game client would send packets to the server at a much higher rate. As a countermeasure, the server may be able to detect the continuously abnormal communication.

Terra's game server seems to send information of other players' locations to the client regardless of whether the client can see those players or not. On a rogue client, this means that the attacker can see more information they should be able to see. One way to prevent this is to only send the client information about objects that are in its visual field. Doing so would conform to the principle of **least privilege**.

F. Denial of Service and Countermeasures

Denial of service in our examples simply comes as a consequence of banning being misapplied on the system. System administrators should **question the assumption** that hackers are the only ones that use the IP range from which they attack. In this case, the entire UBC IP range was banned, and it is not improbable for legitimate players to have been playing at UBC. As well, the assumption that users who cheat are malicious is simply not true in this case. The best way to prevent this denial of service happening would be to maintain account integrity by protecting against password disclosure. As well, it might help to employ **open design** and consult the user community in setting anti-cheating policies, which have provisions for recovery of accounts in case of hijacking.

In the end, we still believe that the best defense against this particular denial of service situation is to provide **defense in depth** by securing the system against the prohibited cheats and attacks. The system would be inherently hardened and administrators would not need to worry about wrongly banning legitimate users.

IV. Conclusion

Throughout the course of this project, we found all of the CIA properties to be at risk. Confidentiality was reduced as a result of the snooping and dictionary attack vulnerabilities. Integrity was reduced when speed hack was employed to

perform unauthorized movements in the game. Availability was compromised when we discovered a method to get an individual user banned and when an IP range ban was placed on the entire UBC network.

The recommended countermeasures include: transporting credentials over a secure channel; using exponential back-off, jailing, or momentary disabling; removing the 8 character username/password limit; closing all but essential ports on firewall and servers; and performing more action validation on the server-side rather than the client side.

An asset/threat analysis revealed that the most significant asset in games is probably the user experience. If a game is made unmemorable, the game studio's income suffers. As well, the disclosure of users' credentials can have enormous impacts on both that user and the game company. These findings are not only applicable to Terra; any online game should be aware of these issues. Furthermore, the online software community in general should be acutely interested in the security of online games as other types of programs move toward a massively online, distributed model.

We believe that our contribution to the field of computer security with this project has been the discovery of additional techniques, like the mail-server hack, as well as promote ideas as to how one can exploit a commercial web server that hosts an online user game with an increasing customer base. We have shown that web servers that host online virtual worlds do need to be thoroughly tested before they are deployed and simple acts like leaving the username and password unencrypted can potentially lead to the discovery of other major flaws.

ACKNOWLEDGMENT

We would like to thank Professor Konstantin Beznosov of the UBC Electrical and Computer Engineering department for his assistance in helping us get in touch with the administrators from Terraoutlands and providing us with reading material that helped us in this project. We would also like to extend our gratitude to Dale Jackman for being supportive, responsive and considerate of our requests like providing logs, removing bans, etc.

REFERENCES

[1] Hoglund, Greg and McGraw, Gary. Exploiting Online Games: Cheating Massively Distributed Systems. Boston, MA: Pearson Education, Inc., 2008.

[2] Florencio, D. and Herley, C. “A large-scale study of web password habits,” In Proceedings of the 16th international Conference on World Wide Web (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07. ACM, New York, NY, 657-666.

[3] Yan, Jeff and Brian Randell. "A Systematic Clasification of Cheating in Online Games." NetGames 05. New York: ACM Press, 2005.

[4] Wittmann,Art. “The Fastest-Growing Security Threat”. InformationWeek. ABI/INFORM Global,2009.

[5] Stamp,Mark. Information Security: Principles and Practice. Hoboken, New Jersey: Wiley-Interscience- John Wiley & Sons, Inc. 2006.

[6] Anderson, Ross. Security Engineering: A Guide to Building Dependable Distributed Systems.1st Edition. Wiley, 2001.