Single-Player Capture The Flag

Evan Gallo, Benjamin Kaplan, Jason Kuster, and Nathan McKinley

April 30, 2012

Contents

1 Introduction

Capture The Flag, in a security context, is a competition used to train people to identify and exploit vulnerabilities in an appliction. In the original game, two teams were pitted against each other. The flag was a file on a computer, and the last one to "touch" the file was currently in control of it.

More recently, a single-player variant has arisen. This variation is more of a test of personal ability rather than a group competition. In this variant, the player is given access to an account on a computer. Their job is to find a vulnerability that will let them get the password to the next account down the line. They win when they complete all of the challenges set before them.

2 Our Game

This project is a single-player implementation, focusing on vulnerabilities common to networked code. The game is run on 4 different machines. Specifically, we have designed this to run in Amazon's Elastic Cloud Compute (EC2) service.

The game consists of 8 challenges spread out over 4 machines. Rather than using passwords, all of the accounts on the machines are set up using private key authentication. At the beginning of the game, the user is given a file that contains encrypted versions of all of the private keys. Each key is encrypted with a 20-character random password. The goal of the game is to read the text file in the home directory of the final user. There is a README file in every user's home directory that gives hints on how to proceed.

3 The Levels

3.1 Level 1

Vulnerability UNIX Path Hijacking

Summary The goal is to read the password.txt file in the level02 home directory. The user is given access to a small program, as well as the C code for the utility. The program uses setuid() to set the user id to level02 and then uses the system function to call date.

Solution Simply create your own executable program called "date" that reads the password file. Put this file first on the path. Then, run the printingdates program to get the password.

3.2 Level 2

Vulnerability Poor Encryption

Summary The user is given access to a file named file.txt which appears to be gibberish. It is encrypted using a simple letter-replacement cipher. The location of the next machine is stored in the ciphertext.

Solution Use a tool that analyzes the frequency of characters (such as a cryptogram solver) to decrypt the ciphertext. The cleartext is the Declaration of Independence, with the hostname of the next server appended to the end.

3.3 Level 3

Vulnerability Identifying Unsecured Remote Access

Summary The user is given the hostname of the next computer. Upon scanning the server with nmap, an open port is detected. By connecting to this server, it is possible to find the password to the next level.

Solution Connect to the server on port 3210 with telnet, at which point you will be presented with a blank prompt. The commands "ls", "cd", and "cat" work. By listing the contents of the directory, a folder called "secrets" is found, and changing directory and listing again reveals a file called "password.txt". "cat" this file for the password to the next level.

3.4 Level 4

Vulnerability SMTP forging

Summary Through a readme, the user is informed that there is a server using a new ultra-secure email-based password recovery system. All they need to do is send an email from the user account on the local machine to password-recovery and it will reply to their email with their password.

Solution The trick here is that a reply is different than sending a new message. Replies by convention use the Reply-To: header in the Data section to determine the receiving email address.

The mail server is not running on the normal port, but an nmap will tell you where it is. Telnet to that server. In the MAIL FROM: field, put "level04@computer-name". In the RCPT TO: field, put "password-recovery@computer-name". In the Data section, set a Reply-To: header to an email address you control. Send them message, wait a bit, and then check your email.

3.5 Level 5

Vulnerability Packet Sniffing

Summary Through a readme, the user is informed that this machine is an authentication server for another user who is bad with passwords. This user will send their password (known to be a bad password) to the server and the server will respond with the password the user should use on other services (a presumably good password).

Solution The user needs to capture an instance of this exchange and analyze it by hand. They will do this using the packet capture utility installed on the machine, which we have altered to allow for limited users to access hardware directly. Upon doing this, the user will discover the plaintext password ”superdoublesecretpassword” sent to port 9125 on the current machine, and an immediate reponse containing only the password for the next level, sent back over the same connection.

3.6 Level 6

Vulnerability Not-So-Poor Encryption

Summary The user is given access to another file.txt which, again, appears to be gibberish. This time, it is encrypted using single DES, which is known to be a weak encryption algorithm. The password to the next level is stored in the ciphertext.

Solution There are two possible solutions to this problem. One would be to attack DES itself. DES is a weak algorithm and can be broken, given enough time. The other would be to perform a brute-force attack on the key. This proves to be the easier attack, as the key is only four alphanumeric characters.

3.7 Level 7

Vulnerability SQL Injection

Summary The user is informed that there is a URL-shortening service installed on the current machine. It is an FCGI app in python which, depending on the querystring, will either shorten a URL or redirect to a URL already shortened. The app is backed by a mySQL database where the pairs are stored. The strings that the user puts into the query string are not escaped; that’s meant to be done by a frontend.

Solution The user needs to figure out that the SQL statement being used is of the form ”SELECT * FROM urls WHERE longurl=’userstring’”. Since the app only acts on the first result returned, the user needs to ensure that the SELECT call returns nothing, then make sure that the first response to the whole statement they write is meaningful, or nothing will happen. One possible solution is

```
' UNION SELECT * FROM passwords WHERE pw LIKE '%
```

This statement will select anything where the longurl parameter is empty (nothing) and take the union of that result with every result from the table passwords. The password will be stored in the "obfuscated URL" returned.

3.8 Level 8

Vulnerability Buffer Overflow

Summary The user is presented with the source of a program and an executable copy of it. The program contains 4 functions. One of them, which reads the text file to you, is private. The other 3 can be acceessed by passing a number from 0 to 2 to the program

Solution The four functions are put into a struct. All memory in a struct is contiguous. By entering -1 into the program, you access the function located before the start of the array, which is the private function. The private function uses setuid so it's able to access the protected file

4 Setup

Someone wishing to use our game has a large number of options available to them. We have provided three different ways to set up the game, in addition to manual assembly.

Regardless of which method you use, you'll have to run several of the network services before the game starts.

Computer 2 Run app.py

Computer 3 Run mailserver.py

Computer 4 Make sure that app.py is running as an FGCI module in Apache and that MySQL is running.

4.1 Block Device File

We've provided 4 files containing perfect images of the hard drives of the machines in question. If you have local machines and you would like to put these images on them, prepare a partition and use the dd utility to copy the images onto the disks. They should be bootable.

4.2 Filesystem Tarball

We've also provided a complete set of all the files on the disks in machine1 - machine4.tgz. Extract those into a new partition and you should be good to go.

4.3 Amazon AMI

We've also placed the images for these machines for public availability on Amazon's Elastic Cloud Compute service. You can spin up one instance of each by searching for EECS 444 CTF.

4.4 Manual Configuration

Since there are a few places where the locations of the next machine are hard-coded in (the decryption levels, for instance), it might be easier to manually configure the levels. Each level requires a user, a private-public keypair, and authorization to log in without password authentication using the private key. The following steps are also necessary for each level:

4.4.1 Level 1

Password.txt must contain the password for level2's private key, and must be set to permissions 400 for owner level2.

4.4.2 Level 2

The result of Level 2 is not a password. It's the URL of the second machine. It should be written out in words and added to the end of a reasonably long document (we used the Declaration of Independence). Use a cryptogram generator to generate an "encrypted" version of the text. put it in a file in the Level02 home directory.

4.4.3 Level 3

Level 3 runs on the second machine. It consists of a remote pseudo-shell that only allows the user to list the directory, cd, and cat. Run the service on the remote server and then put the password file in a directory that the user running the server can read.

4.4.4 Level 4

The password in mailserver.py must be the password for level5's private key. The mailserver script must be run. The script requires Python 2.6 or 2.7.

4.4.5 Level 5

The service must be started on the machine for level5, preferably by a privileged user, then disowned. The client must be started on the machine for level6, preferably by a privileged user, then disowned. vulnerable_server.py must be modified to send the correct password for level6.

4.4.6 Level 6

Use a DES encryption program to encrypt the password with a 4-character alphabetical key

4.4.7 Level 7

We need an apache server, a mySQL database, the python-fcgi module and the apache fcgi module. The mySQL database must contain a database called "shortener" with the tables "urls" and "passwords". "urls" must contain two string fields, longurl and short. "passwords" must contain "pw" and "host". App.py must have its handler set to FCGI-Script in the apache configuration file.

4.4.8 Level 8

Compile the program and set the setuid bit as root so it can setuid(0) in the program. Make a /youwin.txt file at the root of your drive. It can contain whatever you want, it just has to exist.

5 Playing the Game

If you want to play, the keys are here. Do remember to extract the tarball with the 'p' option; permissions of 400 are required or the EC2 instances will reject the keys. The first server is located at ec2-23-22-4-34.compute-1.amazonaws.com, and the password for the first key is s])FBt4ueEe2Nq?9Cdoe. The keys are encrypted using the Rijndael block cipher (also known as AES) using ccrypt 1.9. ccrypt can be obtained here, and most linux distributions have a package for it in the package manager. The usernames proceed as level0n, where n is the level number. Therefore, the following commands will log you in as the first user:

```
ccrypt -d level01.key.cpt <enter password at prompt>
ssh -i level01.key level01@ec2-23-22-4-34.compute-1.amazonaws.com
```

6 Distribution

If you want to set up your own servers, the files for each individual level are here.

If you want to use our AMIs, they are available as the following AMI IDs, in order: ami-3c72a855, ami-3272a85b, ami-0c72a865, ami-0272a86b.

If you want to see the tarballs of the drives, they are available here: M1, M2, M3, M4.

If you want the actual disk images themselves, they are available here: M1, M2, M3, M4.