Vulnerability Insight - Remote Code Execution

Brian Fajardo
Department of Computer Science
Montclair State University
Montclair, United States
fajardob1@montclair.edu

Obinna Ezeadum
Department of Computer Science
Montclair State University
Montclair, United States
ezeadumo1@montclair.edu

German Guzman
Department of Computer Science
Montclair State University
Montclair, United States
guzmang3@montclair.edu

Abstract—Software vulnerabilities are a persistent challenging risk factor that is ubiquitous. It is only upon successful exploitation of a vulnerability that an attacker can cause greater harm and essentially create opportunities for further harm by either using the same attack vector or combining this attack vector with another in hopes of fulfilling their goal. This brings us to consider, how do we ensure that vulnerabilities do not become threats and are exploited? To answer this question it will require a persistent analytical approach which will be the main focus of our paper. In particular we will analyze 23 instances of Remote Code Execution (RCE) vulnerabilities finding root causes of this vulnerability type. The methodology used to analyze this data was to use industry standard resources such as the National Institute of Standards and Technology (NIST) National Vulnerability Database (NVD), The MITRE Corporation Common Weakness Enumeration (CWE), and Common Vulnerabilities and Exposures (CVE) also provided by MITRE to name a few. By carefully examining various RCE root causes, the vulnerability landscape can be reduced providing security professionals meaningful insight into how to prevent this vulnerability type from being exploited.

I. INTRODUCTION

From lack of visibility in both the Software Development Life Cycle (SDLC) and IT infrastructure, software vulnerabilities present the greatest risk factor to almost anyone who does not have a transparent and firm control on their software security, and IT infrastructure. This risk factor can become very problematic very quickly when left unnoticed and no action is taken to eliminate the vulnerability. What makes a vulnerability so problematic then? A vulnerability as defined by The MITRE Corporation is, "A flaw in a software, firmware, hardware, or service component resulting from a weakness that can be exploited, causing a negative impact to the confidentiality, integrity, or availability of an impacted component or components."[1] The words weakness and exploited go hand in hand here. If a vulnerability is not exploited successfully then the risk factor is still present but the risk is lower! In other words you can have a weakness present but the attack vector is reduced. However the main dilemma that exists is how can we ensure a vulnerability is not exploited. The definition of a vulnerability is important to understand here. Although unsuccessful exploitation of a vulnerability does not increase risk, risk is still present! This is why vulnerability and exploit go hand in hand. One depends on the other to satisfy a certain requirement. This idea served as motivation for understanding how a vulnerability can still pose a risk even when it is not exploited. The main dilemma becomes even more challenging when the issue at hand is ubiquitous. Software is constantly being developed. Almost every industry relies on some software for their needs. Software Developers can introduce vulnerabilities and they may not even notice it. This is why it is important to emphasize how impactful software vulnerabilities can be and the importance of having a transparent and firm control on this issue. The vulnerability type that we analyzed is RCE or Code Injection is, "The general term for attack types which consist of injecting code that is then interpreted/executed by the application. This type of attack exploits poor handling of untrusted data. These types of attacks are usually made possible due to lack of proper input/output data validation."[2] RCE vulnerabilities exist in many applications and can have different weakness types associated with it. One analysis we found was on the BlueKeep vulnerability.[3] This instance of vulnerability was discovered initially to have a Denial of Service (DOS) attack vector. However after several months of research and further analysis by other security professionals, it was discovered that BlueKeep could be turned into a RCE.[4] This is one of the first 23 instances that was analyzed and provided us an early solid walkthrough of vulnerability analysis. This related work can be applied to real world vulnerability analysis which is a common security process that both small and large enterprise-sized companies employ. We

can apply this process in a future security role where not only do vulnerabilities need to be identified and analyzed, but also prioritized based on their risk factors.

II. METHODOLOGY

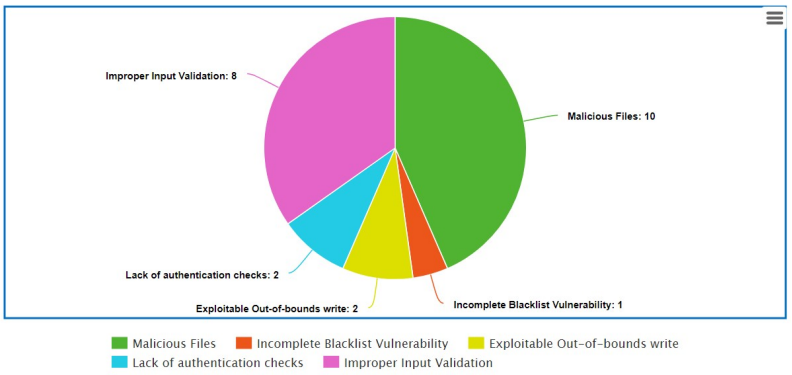
A. Systems Statistics: The systems that were worked on were Microsoft Windows NT family of operating systems versions (XP, Server 2003, Vista, Server 2008, and 7). Other systems included Microsoft Excel versions 2010 to 2019, Windows Media Audio Decoder windows versions (7, Server 2008, Server 2008 R2, Server 2008 Server Core Installation, 8.1, RT 8.1, Server 2012, Server 2012 R2, Server 2012 Server Core Installation, Server 2012 R2 Server Core Installation, Server 2016, Server 2016 Server Core Installation, 10, and lastly server versions 1903 Server Core Installation, 1909 Server Core Installation, Server 2019, Server 2019 Server Core Installation, and Server version 2004. Other systems included Cisco Small Business 300 Series (Sx300), MiniShare 1.4.1, Foxit Software (Studio Photo 3.6.6.913, Phantom PDF 9.7.0.29455, Foxit Reader 9.7.0.29478), ImageMagick 7.0.1-0 / 6.9.3-9, Nagios-NRPE 2.14, and F5's BIG-IP Traffic Management User Interface 16.0.0 and below. Each of these systems that were studied had different years in which the associated vulnerability was disclosed for the respective versions. The years ranged from 2013, 2016, and 2018 through 2020.

B. Vulnerability Collection: The entire process of collection of vulnerabilities involved investigating and collecting vulnerability histories such as vulnerability name / CVE ID / affected versions / fixed version / and information of the developers. After collecting this information then the next step was to explore the vulnerable files and their related descriptions to understand why the vulnerability happened. A total of 23 instances of RCE vulnerabilities were found. We found that The MITRE Corporation's CVE and CWE sites, as well as NIST's NVD site provided most of the collection of information that we wanted to collect. In addition to these resources CVE Details provided early insight to the specific vulnerability that was being investigated. For example in one of the vulnerabilities it was discovered by surprise how the information reported had not been updated in terms of how the vulnerability progressed over time. This served not only as a surprise but also a challenge since vulnerabilities are known to change like in the case of BlueKeep. This is why we considered to include CVE Details as a resource since we can see the before and after picture of how a vulnerability progresses over time in terms of attack vectors and of course any impactful advisories such as mitigations or patches to eliminate the vulnerability from being exploited entirely. Lastly, many of the investigated vulnerabilities had advisories and analysis either done independently by the affected company's products/systems or cooperatively with other companies/researchers. This proved to be the most beneficial scenario for us since not only did it provide more information because there was a thorough analysis already done but we were able to see whether any root causes could be connected with the other instances of RCE vulnerabilities that were investigated! This helped tremendously with also not having to search endlessly for information from the RCE instance that was being investigated since most of the information was presented to us already.

C. Vulnerability Analysis: Analysis of the vulnerable code to find out the common patterns consisted of several approaches. Reviewing the patch files was one approach we conducted with the instance of BlueKeep. This instance provided a thorough analysis compared to the other instances we investigated. For example the author had provided a complete analysis of why the vulnerability had occurred and even provided a breakdown of the vulnerable code and patch file with screenshots that included pre and post patch of the affected code. We were not able to find such a resource of this magnitude in time with the other instances. However this is not to say that the other resources used were not helpful! It simply means that it provided everything that we needed all from one site and we only used a few related resources to expand on what was being explained about BlueKeep to make sure the information was correct across resources. Other approaches included analyzing the advisories which included descriptions of the vulnerable code with some advisories linking to the patch file itself. This approach made up the majority of instances we found. For example out of the total of the 23 instances, 22 instances were analyzed using this approach compared to BlueKeep where we dug deep to analyze the vulnerable code and see how the patch file fixed the affected code. These two approaches, although different, still provided the necessary information that we ultimately sought which was to find root causes of RCE vulnerabilities to support us in our dilemma.

III. RESULTS

A.Findings:



In our findings, we found that the two most common root causes into an RCE vulnerability were from an improper input

validation error or downloading malicious files; 35% and 43% of our collection of root causes had these. There are many ways these root causes could perform an RCE exploitation.

For example a common way that improper input validation is done is by the uses of functions eval(), system(), and exec() in PHP code. These functions are known for having sanitization issues which in turn lead to an RCE[5]. For example:

```
<?php

$cmd=$_GET['cmd'];

 echo exec($cmd);

?>
```

This code can easily be exploited with a simple code injection; however, it is very impractical and not used in most real-life cases unless the coder is very inexperienced.

In a close to real-life example, this code can be exploited:

```
 $ns  = $_GET['ns'];

 system ("dig @$ns $host $query_type");
```

In this code example, the variable $ns is unfiltered and can be injected with the following code command:

"dig.php?ns=||whoami||&host=sirgod.net&query_type=NS&status=digging".

When downloading malicious files, an attacker can also perform an RCE. An attacker can trick a victim to download a file through email. These emails are usually in the forms of system updates. One notable instance of an RCE performed via a malicious file download was the 2017 WannaCry ransomware. This attack was made possible by sending crafted messages to vulnerable SMB ports and using DoublePulsar to install the WannaCry malware. DoublePulsar is described to be the 'backdoor' installed on compromised computers [14]. Once downloaded, WannaCry can perform arbitrary commands that would hold your data on your computer as hostage in return for crypto currency. This is done by encrypting your data, so you are unable to read some valuable files or locking you out of your computer completely.

WannaCry ransomware note displayed on exploited systems [21].

In our research, we found that there's a big leap towards finding an RCE vulnerability to exploiting the RCE. Such as finding an RCE that runs commands, there would be different commands based on the system (Linux or Windows). A hacker would have to customize these commands. One would have to write custom exploits, sometimes they are only found in obscure software on one endpoint. To write an exploit, you need to understand the vulnerability; what causes it, why it happens, what is happening on the server side. A hacker must go through stages like writing code, improving it, and getting it to work on the server side. To find some ways to exploit an RCE some hackers chain vulnerabilities. Chained vulnerabilities are vulnerabilities that have been linked together to become more than their counterparts. In the search for common root causes, there was an instance where an improper input validation could lead to sending a link that held malicious code. The malicious file could then lead to an RCE [22].

IV. DISCUSSION

We can see in the results of our findings that out of the 23 RCE instances, the root causes from most common to least common are 10 instances of malicious files, 8 instances of improper input validation, 2 instances one from lack of authentication checks and 2 instances from exploitable out-of-bounds write, and lastly 1 instance of incomplete blacklist vulnerability. These common patterns have shown that that both the attack vector and weakness provides clarity as to determining the root causes. For example we learned that in our BlueKeep instance how the root cause was due to improper input validation resulting from a use-after-free vulnerability with an identification of CWE-416 [19]. The

Common Weakness Enumeration site provided the weakness types that are associated with vulnerabilities that are disclosed. The approach was to correlate the weakness with the specific RCE instance that was being studied and from there compare it with the other RCE instances [19] [20]. This provided concrete evidence in basing our findings for the root causes and patterns associated with RCE vulnerabilities.

V. THREATS TO VALIDITY

The shortcomings that we think might hamper our results are that not all RCE vulnerabilities share the same weakness type. We were very encouraged by our findings with BlueKeep CVE-2019-0708. However with CVE-2020-1498 it's weakness type is different which is CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer [23] [24]. We learned that this weakness type is associated with CWE-20: Improper Input Validation labeled as can follow which means that the weaknesses are actually associated. [20] [24]. This actually does support our findings of root causes in RCE vulnerabilities. However the complexity arises when different attack vectors are introduced and also when vulnerabilities have been studied over time since more insight can be discovered that was previously not discovered before when the vulnerability was first disclosed. The main idea to understand here is that one cannot simply argue that most RCE vulnerabilities are caused by improper input validation resulting from a use-after-free vulnerability. One must diligently review many instances and compare the weakness types as well as the attack vectors. Having both of these metrics in mind will serve well for any security professional in making informed decisions!

VI. CONCLUSION AND FUTURE PLAN

It takes a whole village for validating and providing analysis on vulnerabilities. The 23 instances of Remote Code Execution vulnerabilities that were studied showed how important it is to understand why a specific vulnerability type occurs. Although the most common root cause of this vulnerability type is due to malicious files, there are also more root causes that can cause an RCE vulnerability to occur like improper input validation, lack of authentication checks, exploitable out-of-bounds write, and incomplete blacklist vulnerability. The vulnerability landscape for RCE vulnerabilities needs to be approached with a persistent and analytical approach. One must not only rely on advisories but also correlate the weakness types and attack vectors that are associated with each vulnerability type. Having such insight is meaningful in making informed decisions as well as prioritize each vulnerability based on their risk factor. Although not every RCE instance will have the same weakness type, we learned that some weakness types still correlate with the root cause that was found for the associated vulnerability. The future plan would be to always consider having both attack vectors and weakness types in mind when analyzing a vulnerability. These metrics will ultimately reduce the vulnerability landscape and prevent the vulnerability from being exploited!

REFERENCES

[1] "CVE -Terminology", Cve.mitre.org, 2020. [Online]. Available: https://cve.mitre.org/about/terminology.html. [Accessed: 05- Dec- 2020].

[2] "Code Injection Software Attack | OWASP Foundation", Owasp.org, 2020. [Online]. Available: https://owasp.org/www-community/attacks/Code_Injection. [Accessed: 06- Dec- 2020].

[3] "Analysis of CVE-2019-0708 (BlueKeep) - MalwareTech", MalwareTech, 2020. [Online]. Available: https://www.malwaretech.com/2019/05/analysis-of-cve-2019-0708-bluekeep.html. [Accessed: 06- Dec- 2020].

[4] "BlueKeep: A Journey from DoS to RCE (CVE-2019-0708) - MalwareTech", MalwareTech, 2020. [Online]. Available: https://www.malwaretech.com/2019/09/bluekeep-a-journey-from-dos-to-rce-cve-2019-0708.html. [Accessed: 06- Dec- 2020].

[5] "How to find RCE in scripts (with examples)", Exploit Database, 2020. [Online]. Available: https://www.exploit-db.com/papers/12885. [Accessed: 06- Dec- 2020].

[6] N. Ermishkin, "ImageMagick 7.0.1-0 / 6.9.3-9 - 'ImageTragick ' Multiple Vulnerabilities", Exploit Database, 2020. [Online]. Available: https://www.exploit-db.com/exploits/39767. [Accessed: 07- Dec- 2020].

[7] "Security Update Guide - Microsoft Security Response Center", Msrc.microsoft.com, 2020. [Online]. Available: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2020-1508. [Accessed: 07- Dec- 2020].

[8] "Security Update Guide - Microsoft Security Response Center", Msrc.microsoft.com, 2020. [Online]. Available: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2020-1498. [Accessed: 07- Dec- 2020].

[9] O. Whitehouse and V. Whitehouse, "Understanding the root cause of F5 Networks K52145254: TMUI RCE vulnerability CVE-2020-5902", NCC Group Research, 2020. [Online]. Available: https://research.nccgroup.com/2020/07/12/understanding-the-root-cause-of-f5-networks-k52145254-tmui-rce-vulnerability-cve-2020-5902/. [Accessed: 07- Dec- 2020].

[10] "Security Bulletins | Foxit Software", Foxit, 2020. [Online]. Available:

https://www.foxitsoftware.com/support/security-bulletins.html. [Accessed: 07- Dec- 2020].

[11] "ImageTragick", Imagetragick.com, 2020. [Online]. Available: https://imagetragick.com/#:~:text=There%20are%20multiple%20vulnerabilities%20in,being%20used%20in%20the%20wild. [Accessed: 07- Dec- 2020].

[12] "Remote code execution in Nagios Remote Plug-In Executor (NRPE)", Cybersecurity-help.cz, 2020. [Online]. Available: https://www.cybersecurity-help.cz/vdb/SB2013070909. [Accessed: 07- Dec- 2020].

[13] 2020. [Online]. Available: https://support.f5.com/csp/article/K52145254. [Accessed: 07- Dec- 2020].

[14] "What is WannaCry ransomware?", usa.kaspersky.com, 2020. [Online]. Available: https://usa.kaspersky.com/resource-center/threats/ransomware-wannacry. [Accessed: 07- Dec- 2020].

[15] "Create A Bar Chart, Free . Customize, download and easily share your graph. Just enter the amounts, pick some colors/fonts, and we'll take it from there!", Meta-chart.com, 2020. [Online]. Available: https://www.meta-chart.com/bar. [Accessed: 07- Dec- 2020].

[16] "RDP : Stairway to Networks - K7 Labs", K7 Labs, 2020. [Online]. Available: https://labs.k7computing.com/?p=19194. [Accessed: 07- Dec- 2020].

[17] "NVD - CVE-2019-0708", Nvd.nist.gov, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-0708#match-4280312. [Accessed: 07- Dec- 2020].

[18] "CVE-2019-0708 : A remote code execution vulnerability exists in Remote Desktop Services formerly known as Terminal Services when an unau", Cvedetails.com, 2020. [Online]. Available: https://www.cvedetails.com/cve/CVE-2019-0708/. [Accessed: 07- Dec- 2020].

[19] "CWE - CWE-416: Use After Free (4.2)", Cwe.mitre.org, 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/416.html. [Accessed: 07- Dec- 2020].

[20] "CWE - CWE-20: Improper Input Validation (4.2)", Cwe.mitre.org, 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/20.html. [Accessed: 07- Dec- 2020].

[21] A. Chiu, "Player 3 Has Entered the Game: Say Hello to 'WannaCry'", Blog.talosintelligence.com, 2020. [Online]. Available: https://blog.talosintelligence.com/2017/05/wannacry.html. [Accessed: 07- Dec- 2020].

[22] "Cisco Small Business 300 Series Managed Switches Authenticated Reflected Cross-Site Scripting Vulnerability", Tools.cisco.com, 2020. [Online]. Available: https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20180801-sb-rxss. [Accessed: 07- Dec- 2020].

[23] "NVD - CVE-2020-1498", Nvd.nist.gov, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-1498. [Accessed: 07- Dec- 2020].

[24] "CWE - CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer (4.2)", Cwe.mitre.org, 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/119.html. [Accessed: 07- Dec- 2020].