Reliability of exploits and consequences for decision support

Maxwell Dondo and Jonathan Risto
Defence Research and Development Canada
Ottawa, Canada
maxwell.dondo@drdc-rddc.gc.ca
jonathan.risto@drdc-rddc.gc.ca

Reginald Sawilla
NATO Communications and Information Agency
The Hague, Netherlands
reginald.sawilla@ncia.nato.int

ABSTRACT

Limited public information is available about the actual likelihood of success that attackers will have when attempting to exploit a particular vulnerability. The metrics that are available are therefore used to meet the demand for this type of information but that usage does not lead to an accurate threat picture. The exploitability of specific vulnerabilities depends upon the network environment and the attacker of concern, thus there is no reason to expect that metric information that does not include these attributes in its scope will lead to a correct mitigation prioritization, even if that metric information is correct within its scope. However, insufficient threat information, or an incomplete understanding of the scope of particular metrics, leaves network defenders to use the metrics they have for purposes outside of their scope, and that can cause network defenders to prioritize mitigations inappropriately.

In this paper we model the largest class of attackers – a basic attacker who uses the widely available Metasploit Framework (MSF) penetration testing tool with its dictionary of exploits. We show that there is only a moderate relationship between the popular Common Vulnerability Scoring System (CVSS) exploitability metric, which provides an indication of the exploitability of a vulnerability, and the success of an attacker in our attacker model. In environments where resources are constrained so that vulnerability mitigation must be prioritized, this work demonstrates that an efficient use of resources will not be obtained by relying on public vulnerability metrics alone. It is important to determine the environmental exploitability of vulnerabilities by testing the attacker models of concern against the organization's installed baseline, and prioritizing mitigations according to success likelihood data. We present an application of our approach and provide statistics from testing across a wide range of Microsoft operating systems.

1.0 INTRODUCTION

Computer software vulnerability advisories contain information about the privilege that attackers would gain if they exploited a vulnerability, but they do not contain information about how reliably attackers can execute an exploit against a vulnerability. In other words, network administrators do not know whether attackers can in fact successfully exploit the vulnerability. This information is an important input to allow administrators to prioritize their limited resources in mitigating one vulnerability over another. Thus, they are left to assume that the presence of vulnerable software in their network is sufficient to enable attackers' successful exploit of the vulnerabilities and they prioritize their mitigation activities based on insufficient metrics from information sources that lack details on the likelihood of successful exploitation of the vulnerabilities.

If mitigation of vulnerabilities could be done quickly, inexpensively, and with no impact on the enterprise, then one could simply mitigate all of them. The reality is that there are many more vulnerabilities on enterprise-sized networks than can be addressed with affordable levels of resourcing, and within acceptable levels of impact. Without quality data of sufficient detail on vulnerabilities, the scarce resources will not be spent efficiently, and the network will be less secure than it would be if the resources were spent optimally by mitigating the vulnerabilities that are the most likely to be exploited.

The focus of this work is to test the hypothesis: software vulnerabilities are easily and reliably exploitable. We show the hypothesis is false and that vulnerabilities are often not easily and reliably exploitable. We develop a repeatable approach to determine the likelihood of successful exploit of vulnerabilities in a given attack model. Attack graphs can use our success likelihood metric, and the dependence that an attacker has on the vulnerabilities, as an input to derive mitigation priorities to efficiently defend a network. We compare our prioritization with that obtained if one were to simply use published vulnerability scores. We present an application of our approach using the attack model of a person using the Metasploit framework, and give statistics from data gathered across many software platforms. We show that the success likelihood varies greatly across exploits. In addition to improving mitigation prioritization, the research is also directly applicable to threat and risk assessments.

Section 2.0 describes the infrastructure using virtual machines (VMs) that are comprised of an attack station, running the attack tool MSF, and separate target machines running the Operating Systems (OSs) under investigation. The setup enables the execution of attacks against one targeted OS at a time, allowing us to collect relevant statistics for each OS as described in Section 3.0. This is followed by related work in Section 4.0. Discussion of our results and conclusions are presented in Section 5.0.

2.0 DESCRIPTION OF TEST INFRASTRUCTURE

In this section, we describe the logical and physical setup of the test environment. We present the network architecture, routing, and system configurations. The section is concluded with a presentation of our attacker model and the test tools used.

2.1 Testing environment and Network architecture

Our testing was conducted in a lab environment using virtualization technology to emulate a penetration testing scenario. Virtualization technology, in the form of VMs, was used to simplify the testing infrastructure. Attacker and target systems as well as the connecting switch were all created within the VM environment. The VM network was isolated to ensure attacks to targets would not spread to unintended systems or networks. Virtualization also permitted the automation of some components of the testing, such as automating the power cycling, and ease of configuration, deployment and control.

In this testing environment, all devices were placed into a flat network. Target systems were placed into the same network segment as the attacker systems. This general architecture is illustrated in Figure 1. It consists of n hosts directly connected to the N attacker systems through a virtual network switch. During our testing, one of the attacker systems was used to launch a direct attack on one of the targeted hosts. This configuration represents a single step attack, which conforms to the type of attacker represented by our tests. In addition, antivirus products and host firewalls were disabled in order to ensure that our experimental results were independent of any security configurations on the target systems. While not representative of a production deployment of systems, this setup helped ensure that the success and failure rates of the exploits used were not externally influenced by security controls.

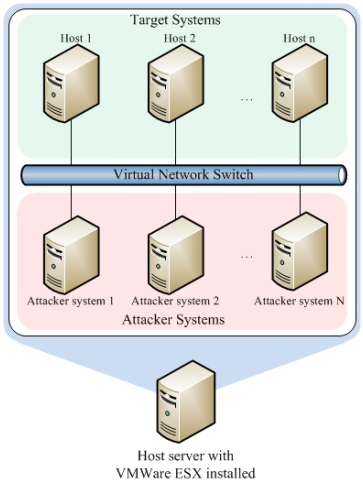

Figure 1: Generalised test network architecture.

2.2 Attacker model and testing tools used

During penetration testing using attack tools, the general steps taken to perform attacks are as follows [1]. First, the services available on a target system are determined. Secondly, vulnerabilities on that system are established. Next, an exploit that takes advantage of this vulnerability is delivered to the target using a selected communication mechanism. The exploit is then executed, and access to the exploited system is achieved. The attacker model and the attack tools used to execute these steps in our tests are presented in this section.

2.2.1 Attacker model

Our attacker model represents a determined attacker who uses limited theoretical knowledge and up to moderate practical knowledge to gain access to target systems. The attacker, who can take up to a few weeks to plan and execute attacks, is unwilling to accept any negative consequences of his actions. However, the attacker's use of amateurish knowledge in pursuit of his goals means that he is unable to maintain any level of secrecy beyond that provided by the tools he uses. Once the attacker gains access to the target systems, he is able to perform simple operations, such as exfiltration of easily available information.

This attacker model fits the threat level 8 of the Sandia generic threat matrix [2] in a number of ways. Our attacker uses exploits in the attack tool library to gain a shell and uses limited attack knowledge to pursue his goal, which in our case is exfiltrating easily acquired data in the form of a screenshot. Matching the time it took us to plan and execute the attacks, our attacker takes up to a few weeks to do the same tasks. In addition, during the execution of attacks, our attacker uses and modifies existing attack scripts that targeted vulnerable systems. In those scripts, the attacker exploited vulnerable systems by using the basic payloads provided by the attack tools. All these attacker characteristics, which describe the common hacker, also make our attacker model fit in level 1 of the MITRE Cyber Preparedness model [3].

This model was chosen since the majority of the attacks that organizations face are perpetrated by attackers that fall into this model [4, p. 226]; hackers accounted for the highest data breaches in 2013 [5, p. 39]. Sophisticated multi-stage attacks could be considered as implementing our attacker model multiple times. While we neither discount the existance of more complex and knowledgeable attackers exist, nor the potential threat that they pose, these added layers of complexity are beyond the scope of our work.

2.2.2 Metasploit Framework

Our penetration testing tool, Metasploit Framework (MSF) [6], is a multi-purpose attack suite that provides, among other functionalites, the ability to exploit vulnerable remote systems. Penetration testing is accomplished by using exploits that come with the MSF suite or through custom code running within the framework. For this work, and to match our attacker model, only the exploits that come with the MSF were used and no customization was performed beyond automating the tools to simplify testing.

Besides exploits, the MSF environment contains many payload modules that permit users to deploy code onto remote targeted systems. The payloads are delivered to the target using one of the various delivery mechanisms [6, 7]. To provide connectivity to a remote system, MSF also supports many different methods of transmitting data between the target and attacker [6, 7]. This could include the attacker initiating the connection to the target system, or the target system connecting out to the attacker system. For this work, we utilized the two most popular methods: Reverse TCP and Bind TCP.

Our testing involved delivering the MSF shell called "Meterpreter". Meterpreter provides a commandline shell similar to the Unix or Windows DOS shells. The penetration tester can enter commands through the shell to perform desired actions on the remote target. Detailed information on the functionality of MSF can be found at [6], [7] and [1].

2.2.3 CORE Impact and Nessus

During the course of our testing, another penetration testing tool, CORE Impact [8] was used. CORE Impact is a commercial penetration testing tool available from CORE Security. The objective of using this tool in our testing was to validate the results of our experiments. The attacker in our model would not have access to this tool due to its price point.

Nessus is a vulnerability scanning tool that enables users to find vulnerabilities within a target system. Both MSF and CORE Impact can perform vulnerability scans before launching attacks. In order to validate the vulnerabilities used in this work, we used the commercial version of Nessus to scan the target hosts for vulnerabilities. These results were compared with the findings of the database searches and attack tools.

3.0 DESCRIPTION OF TEST SETUP AND RESULTS

In this section, the testing process is described. Selection of the vulnerabilities that were exploited on specific Windows platforms is discussed and detailed results for each OS are given.

3.1 Test setup

To measure the success rate of exploiting a vulnerability, each OS was repeatedly attacked in two ways: with and without rebooting between attacks. The first experiment was to repeatedly attack specific OSs 100 times. Each attack (successful or not) was allowed to finish before the next attack was commenced. We chose this attack approach to simulate a real-world scenario where various attackers could be attempting to gain access to a system without the users' knowledge and intervention. The repetition helps to demonstrate if any of the attempted exploits broke a service or functionality on the target, the success of subsequent attacks could be affected. The second experiment also repeated the attack against the specific OS 100 times. However, in this scenario, each target was rebooted between attacks. The reboot helped ensure that the host OS was in a "stable" state and that any difficulties in getting the exploit working correctly would be attributable to the exploit code itself.

The use of both the Reverse TCP and the Bind TCP stagers eliminates the possibilities of recording exploitation failures resulting from failed stager code. However, throughout our tests, we concluded that no payload delivery mechanism resulted in better success rates than the other. We therefore recorded the better success rates from the two techniques.

Meterpreter's shell has many commands available to the attacker. One of these commands, `screenshot`, was executed once Meterpreter had been deployed, and captured an image file. This file was then saved on the attacker's computer as proof of a successful attack.

3.2 Selected operating systems

For our tests, we chose a representative sample of OSs that are in use across organizations. Since Microsoft (MS) Windows OSs dominate the market share with over 90% of the desktop operating systems installed [9], we chose them for our experiments. We used unpatched OSs as provided through the Microsoft Developer Network (MSDN). The selected OSs as well as the reasons for selecting them are listed in Table 1.

Table 1: Summary of OSs selected.

Operating system	Reasons for selection
Windows XP (No service pack (SP))	Most popular OS constituting over 43% of deployed OSs in 2012 [9].
Windows XP with SP3	Most recent version of Windows XP available from MSDN. This version was chosen to model an up-to-date version of this popular OS.
Windows Vista with SP2	Successor to Windows XP. This version contains the most recent SP from MS.
Windows 7 with SP1	Had over 40% of the market share in 2012 [9].
Windows Server 2003 with SP 2	Representative server software deployed within organizations. SP2 contains the most recent roll-up of patches that MS has deployed.
Windows Server 2008 R2 with SP1	The most recent server software provided by MS.

There are recent Microsoft OSs omitted from our testing. At the time of OS selection, the Windows 8 OS was not yet released. Windows Server 2012, while available, had just been released. Windows Server 2000 is no longer supported by MS. Testing the reliability of exploits against an obsolete OS would not be representative of exploit reliability against more current OSs.

3.3 The test network

Our detailed test network is shown in Figure 2. All devices were placed onto a flat network. To simplify scripting requirements during testing, each workstation was assigned the same IP address. During testing, only one target machine was running at any given time. The target OSs listed in Table 1 were loaded on each host as illustrated in Figure 2. MSF and automation scripts are hosted on the first penetration testing system. CORE Impact and the Nessus vulnerability scanning tool are hosted on the other two attacker systems. During testing, only one attacker system is used at a time.

Multiple versions of MSF were configured on the attacker VM, as exploitation problems[1] were found with one version of the software that were not present in older versions. Although the primary testing was performed with MSF version 4.2, versions 3.4, 3.6, 4.0, 4.1.4, and 4.5 were also used for test validation in this part of the experiment.

[1] Some exploits failed in one MSF version, but succeeded in other versions.

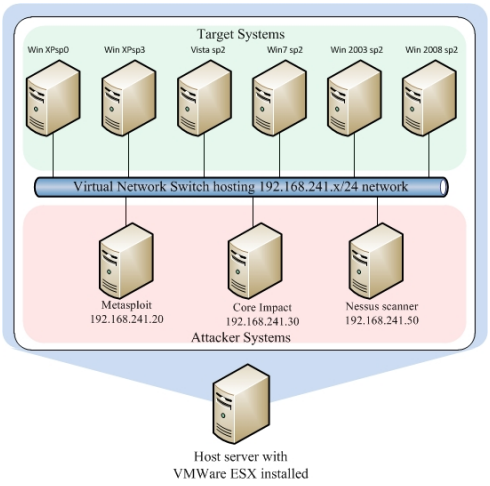

Figure 2: A diagram illustrating the network architecture used in this work.

3.4 Vulnerability and exploit profiles

A list of vulnerabilities was created from vulnerability data available. Our sources of vulnerabilities were the National Vulnerability Database (NVD) [10], the Open Source Vulnerability Database (OSVDB) [11], and the MS Security TechCenter Security Bulletins [12]. From these sources, we selected, to suit our attacker model, server side exploits rather than client-side exploits. Client-side exploits require user interaction, such as a mouse click [13, 14]. Examples of client-side exploits are email-born viruses, such as those that have exploited MS Visual Basic vulnerabilities, which require users to click a link or open an attachment for the attack to succeed. Server-side exploits do not require user input to succeed. Examples of server-side attacks are the system compromise resulting from the remote execution of arbitrary code in worms such as Blaster and Nachi.

Of the $53,378$ general vulnerabilities found within the NVD as of 21 November 2012, 727 had at least one MSF exploit associated with them. Of these vulnerabilities, 135 affect MS software and their dependencies. Since our tests only focus on server-side vulnerabilities, this was further reduced to 9 unique server-side vulnerabilities affecting the MS OSs listed in Table 1. In addition to these vulnerabilities, one vulnerability was identified that was only listed in the OSVDB. Our low vulnerabilities figures only represent the server-side vulnerabilities for MS OSs for which there were MSF exploits. As shown in this report, some vulnerabilities are present in multiple OSs.

3.5 Test results

Using the procedure described earlier, the exploitation process was run against targeted OSs. We then recorded the success and failure rates. In addition to our test result, we listed the CVSS [15] exploitability subscores[2] as recorded in the NVD for each vulnerability [10]. The exploitability subscore is computed from the Access Vector, Access Complexity, and Authentication metrics [15]. CVSS defines the subscore as a measure of how readily exploitable (require low effort) a vulnerability is, ranging from a low value of 1 to a high value of 10 (least effort). We summarise our test results for each targeted OS in this section.

[2]Some OSVDB vulnerabilities, such as OSVDB-397, do not have assigned CVSS scores.

3.5.1 MS Windows Server 2008 Revision 2 Service Pack 1

Two vulnerabilities were identified for MS Windows Server 2008 R2 SP1. Vulnerability CVE-2010-2729, exploitable by the MSF exploit /smb/ms10_061_spoolss, allows remote code execution by sending crafted print requests over remote procedure calls (RPCs) [10]. The other vulnerability OSVDB-397, which is exploitable by the MSF exploit /iis/iis_webdav_upload_asp, allows remote attackers to upload arbitrary files to a web server, which can result in information modification such as the defacing of web sites.

We failed to exploit either vulnerability using the corresponding MSF exploits. In both cases, no Meterpreter session could be established. This failure suggests that users in our attacker model would fail when using such exploits to launch attacks on this OS. However, the CVSS exploitability subscores tell a different story. Although, as stated earlier, there is no CVSS exploitability subscore for vulnerability OSVDB-397, CVSS assigns a value of 8.6 on vulnerability CVE-2010-2729. This high CVSS score points to a easily exploitable vulnerability. However, our experimental results, using a publicly available MSF exploit do not support that.

The failure of exploit /smb/ms10_061_spoolss is attributed to the failure to execute the printer exploit code on the target. The execution of the exploit is in two stages. The first stage, which involves the creation of files in a system directory, was successful and the files appeared in the printer queue. The second stage of the exploitation involves the execution of the exploit when the printer queue processes the exploit files in the print queue. However, attempts to execute the exploit failed, and no MSF Meterpreter session was established.

Exploit /iis/iis_webdav_upload_asp failed because the Internet Information Services (IIS) 7 web server included with Windows Server 2008 would not accept hypertext transfer protocol (HTTP) PUT requests required for exploitation. Unlike Windows Server 2003 and IIS 6, IIS 7 no longer supports the ability to perform anonymous PUT requests by default. Although effort was made to configure IIS 7 to accept HTTP PUT verbs at the IIS server and site group levels, the changes required significant configuration modifications to the IIS server, custom modules and configurations to be created, which was a significant deviation from a "standard" web server configuration. Thus, although vulnerability literature indicates that this OS is vulnerable to this exploit, our research indicates the vulnerability has dependencies on optional subcomponents of the OS, resulting in the failed exploit.

3.5.2 MS Windows 7 SP1

No server side vulnerabilities with MSF exploits for MS Windows 7 SP1 were identified. This unlikely result suggests that the OS with the second widest market share [9] has no server side vulnerabilities that can be exploited through commonly available tools such as MSF. This result points to the importance of having an exploitability measure for every vulnerability for a given OS, so that analysts can optimally allocate their resources to efficiently respond to known vulnerabilities.

3.5.3 MS Windows Server 2003 SP2

We identified 3 applicable server-side vulnerabilities for MS Windows Server 2003 SP2. The first two vulnerabilities, CVE-2010-2729 and OSVDB-397 are the same as those identified earlier for MS Windows Server 2008 Revision 2 (R2) SP1. The third vulnerability CVE-2008-4250 allows attackers to remotely execute code on the system using crafted RPC requests, thus allowing the system to be completely compromised. The Conficker worm, for example, used this vulnerability to propagate.

Table 2 summarizes the results of the tests. We succeeded in exploiting two of the vulnerabilities and failed to exploit the third. Vulnerability CVE-2008-4250 was exploited 100% of the time with reboots, but only 30% of the time in the "No Reboot" executions. Vulnerability OSVDB-397 was exploited with nearly 100%

consistency. We had only one "No Reboot" failure for this vulnerability. Vulnerability CVE-2010-2729 completely failed to execute.

Table 2: A summary of the exploitation results using MSF on Windows 2003 SP2.

Vulnerability	Exploit	Exploitability subscore	Success Rate	
			No Reboot	Reboot
CVE-2008-4250	/smb/ms08_067_netapi	10	30%	100%
CVE-2010-2729	/smb/ms10_061_spoolss	8.6	0%	0%
OSVDB-397	/iis/iis_webdav_upload_asp		99%	100%

The CVSS subscores for vulnerability CVE-2008-4250 points to an easily exploitable vulnerability. This is supported by our experimental results for the Reboot case. However, the "No reboot" results paint a different picture. The 30% success rate in the more pragmatic "No Reboot" case suggests that the CVSS subscore of 10 tells only one side of the story. The explanation for the failure of vulnerability CVE-2010-2729 is the same as that presented for the Windows Server 2008 R2 SP1 case. The /smb/ms10_061_spoolss exploit failed and, as in the Windows Server 2008 R2 SP1 case, the exploit appears in the shared printer queue but the exploit execution is terminated with a timeout exception.

3.5.4 MS Windows Vista SP2

For this OS, there was one vulnerability with a corresponding MSF exploit. This vulnerability CVE-2010-2729 was explained for Windows Server 2008 R2 SP1. We, however, could not exploit this vulnerability. Identical to the previous two OSs, the exploit appeared in the shared printer queue. In this case, the exploit execution timed-out for reasons we are unable to establish. Its appearance in the printer queue may indicate problems with the MSF exploit code on these OSs.

3.5.5 MS Windows XP SP3

Two server-side vulnerabilities for MS Windows XP SP3 have corresponding MSF exploits. These vulnerabilities, CVE-2008-4250 and CVE-2010-2729, are the same as described in the earlier OSs. We were able to attack the MS Windows XP SP3 OS with success rates shown in Table 3. Without reboots, vulnerability CVE-2008-4250 was exploitable 85% of the time, while vulnerability CVE-2010-2729 was exploitable 99% of the time. With reboots, both vulnerabilities were exploitable 100% of the time.

Table 3: A summary of the exploitation results using MSF on Windows XP SP3.

Vulnerability	Exploit	Exploitability subscore	Success Rate	
			No Reboot	Reboot
CVE-2008-4250	/smb/ms08_067_netapi	10	85%	100%
CVE-2010-2729	/smb/ms10_061_spoolss	8.6	99%	100%

Note that the /smb/ms10_061_spoolss exploit succeeded on this system in contrast to the Windows Server 2008 R2 SP1, Windows Server 2003 SP2, and Windows Vista SP2 OSs where it failed. This reinforces the point that exploits are dependant on the host operating system configuration and patch levels, and illustrates the point that new vulnerabilities can be introduced while correcting older ones.

Both CVSS exploitability subscores are high, indicating vulnerabilities that are readily exploitable. This is supported by our results that show exploitability rates of 85% and higher for both cases. However, our experimental results of 85% success rates in the "No Reboot" case disagrees with the CVSS exploitability subscore that implies that the vulnerability CVE-2008-4250 is exploitable all the time. The CVSS exploitability subscore for vulnerability CVE-2010-2729, points to a vulnerability that is readily exploitable most of the time. However, our experimental results show that this vulnerability is almost always exploitable on this OS.

3.5.6 MS Windows XP SP0

We identified eight relevant server-side vulnerabilities that could be exploited in MS Windows XP SP0[3]. MSF identified nine exploits that could exploit those vulnerabilities. Three of those exploits turned out to be inapplicable to exploit this OS. Vulnerability CVE-2003-0352 can allow attackers to remotely execute arbitrary code through malformed messages as evidenced in the Blaster and Nachi worm attacks. The vulnerability CVE-2003-0533, which is exploited by the Sasser worm, allows remote code execution that leads to unauthorised information disclosure and/or disruption of services. Another vulnerability that could lead to unauthorised information disclosure and/or disruption of services is CVE-2003-0812. Vulnerability CVE-2003-0818 allows for remote code execution. These vulnerabilities and their corresponding exploits are listed in Table 4.

Table 4: A summary of the exploitation results using MSF on Windows XP SP0.

Vulnerability	Exploit	Exploitability subscore	Success Rate	
			No Reboot	Reboot
CVE-2003-0352	/dcerpc/ms03_026_dcom	10	34%	100%
CVE-2003-0533	/smb/ms04_011_lsass	10	34%	100%
CVE-2003-0812	/smb/ms03_049_netapi	10	0%	0%
CVE-2003-0818	/smb/ms04_007_killbill	10	13%	100%
CVE-2006-4691	/smb/ms06_070_wkssvc	10	0%	0%
CVE-2008-4250	/smb/ms08_067_netapi	10	100%	100%

Test results in Table 4 show that we were successful in exploiting 4 of the 6 applicable vulnerabilities. In each one of the 4 successful cases, a 100% success rate was achieved for each "Reboot" exploit. Interestingly, only 34% of the "No Reboot" attacks with the /smb/ms04_011_lsass and /dcerpc/ms03_026_dcom exploits were successful. An even lower 13% success rate for exploit /smb/ms04_007_killbill with the "No Reboot" automation was recorded. To attempt to improve the success rate, all tests were repeated with longer wait times between attacks. Regardless of the length of the wait times, the results remained the same except for /smb/ms04_011_lsass, whose success rate increased from 34% to 100%. This change indicates that the Local Security Authority Subsystem Service (LSASS), which is the service being exploited in this case, takes some time to clean up the system after an attack.

From Table 4, two exploits failed, despite indication from NVD and MSF that they apply to the target OS. MSF indicates that exploit /smb/ms06_070_wkssvc completes execution, but there is no indication of success or failure since we could not get access to the Meterpreter shell. For exploit /smb/ms03_049_netapi, the peer always reset the connection before we had access to the Meterpreter Shell. Other than identifying the failures, we do not have conclusive reasons for them.

[3] SP0 means the original OS without any service packs.

The CVSS exploitability subscores are 10 for all the vulnerabilities for this OS. Except for the two complete failures in CVE-2006-4691 and CVE-2003-0812, our experimental "Reboot" results support the CVSS subscore values. In addition, all our experimental results support the CVSS subscores for vulnerability CVE-2008-4250. The low success rates in the "No Reboot" cases for the rest of the vulnerabilities contradicts the CVSS subscore that point to readily and easily exploitable vulnerabilities. Further contradictions are reflected in the two total failures whose CVSS exploitability subscores are still 10.

3.5.7 Comparison of our success rates with CVSS base scores

CVSS base metrics represent the basic characteristics of a vulnerability. Every vulnerability recorded in the NVD is assigned these metrics. The metrics give an indication of the level of effort required to exploit the given vulnerabilities. For example, a vulnerability with an Access Complexity metric of 0.61 is expected to require more effort to exploit than one with 0.71. Similarly, the exploitability subscore, which is calculated from the first three base metrics [15, Section 3.2.1], also gives a measure of the effort required to exploit a vulnerability. These scores and metrics, as well as our success rates are listed in Table 5[4].

Table 5: A side-by-side comparison of CVSS scores and our success rates.

Vulnerability	Access Vector	Access Complexity	Authentication	Exploitability Subscore	Base Score	Success Rate	
						No Reboot	Reboot
MS Windows Server 2008 R2 SP1							
CVE-2010-2729	1.0	0.61	0.704	8.6	9.3	0%	0%
OSVDB-397						0%	0%
MS Windows Server 2003 SP2							
CVE-2008-4250	1.0	0.71	0.704	10.0	10.0	30%	100%
CVE-2010-2729	1.0	0.61	0.704	8.6	9.3	0%	0%
OSVDB-397						99%	100%
MS Windows Vista SP2							
CVE-2010-2729	1.0	0.61	0.704	8.6	9.3	0%	0%
MS Windows XP SP3							
CVE-2008-4250	1.0	0.71	0.704	10.0	10.0	85%	100%
CVE-2010-2729	1.0	0.61	0.704	8.6	9.3	99%	100%
MS Windows XP SP0							
CVE-2003-0352	1.0	0.71	0.704	10.0	7.5	34%	100%
CVE-2003-0533	1.0	0.71	0.704	10.0	7.5	34%	100%
CVE-2003-0812	1.0	0.71	0.704	10.0	7.5	0%	0%
CVE-2003-0818	1.0	0.71	0.704	10.0	7.5	13%	100%
CVE-2006-4691	1.0	0.71	0.704	10.0	10.0	0%	0%
CVE-2008-4250	1.0	0.71	0.704	10.0	10.0	100%	100%

From Table 5, the CVSS base scores are constant for a given vulnerability regardless of the OS. Consider vulnerability CVE-2008-4250 for example. The CVSS metrics and scores do not change from one OS to another. These measures suggest the same level of effort is required to exploit this vulnerability over all the

[4]Vulnerability OSVDB-397 does not have CVSS metrics or scores, and OSVDB does not have any metrics for it.

vulnerable OSs listed. However, our results show that this is not always the case. Our success rates for this vulnerability vary from 30% to 100%.

Another observation from Table 5 is that all the vulnerabilities recorded have very high base scores. The scores translate to exploitability subscores of 8.6 and 10. These subscores suggest that all the vulnerabilities require little effort to exploit. However, our experimental results show that the success rates vary from 0% to 100%. Those results indicate that the vulnerabilities listed in Table 5 may not always be as exploitable as suggested by the high CVSS base scores.

These score variations are shown in the scatter graph[5] in Figure 3. In the graph, CVSS only has two Access Complexity metrics of 0.61 and 0.71. These values imply that we only have two levels of effort required to exploit the vulnerabilities in Table 5. This observation is partially supported by our "Reboot" success rates which are either 0% or 100%. However, there is only a moderate correlation between our success rate and Access Complexity ($r = 0.47$)[6]. On the other hand, our "No Reboot" success rates show a wide variation with Access Complexity, and its not surprising that there is no significant correlation between the two ($r = 0.14$).

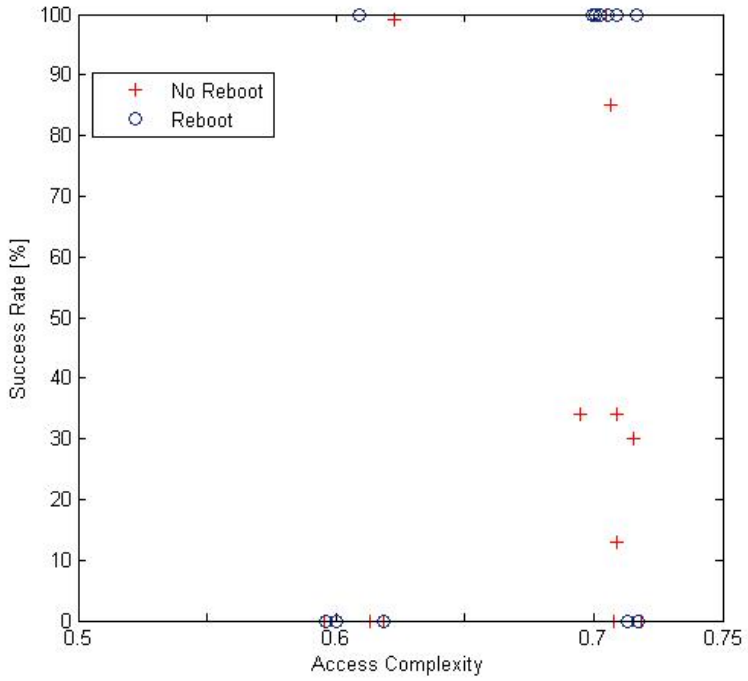

Figure 3: Scatter plot to illustrate the variations in success rates and access complexity.

3.5.8 Validation tests with Nessus scan and CORE Impact

Faced with the seemingly low vulnerability and exploit counts while considering the ever-increasing number of breaches identified [16, 17], we validated our results in two stages. First, the Nessus Vulnerability Scanner [18] was used to validate that the list of exploitable vulnerabilities we found was exhaustive. Then CORE Impact [8]

[5]The graph has a 0.015 jitter.

[6]r is the correlation coefficient.

was used to exploit targeted systems using its default all-inclusive attacks; this verified that the successes and failures recorded earlier in this work are based on an exhaustive list of vulnerabilities for the identified OSs.

Nessus was used to scan each of the OSs under consideration and it identified their vulnerabilities. We compared its list with the vulnerabilities obtained using the NVD and OSVDB databases. Nessus identified the same server-side vulnerabilities as obtained before. This confirmed that our initial vulnerability search was exhaustive.

Each of the OSs under testing were configured as a target system within Core Impact. Attacks were then launched against each of the targets, allowing CORE to attempt to breach the target system. When the attacks for each OS were executed, CORE Impact exploited the same vulnerabilities that we had identified earlier. No new exploits or vulnerabilities were identified that our database searches and MSF testing had not identified previously.

4.0 RELATED WORK

There are a number of approaches to determine some measure of the likelihood of being attacked. We identified four different categories of these efforts and compared them to ours. First, we identified some approaches [15, 19, 20] that focus on quantifying known and unknown vulnerabilities and their attributes as an indicator of how likely a system can be attacked. As our work (which depends on known and published vulnerabilities) shows, without testing the vulnerabilities on given targets, it cannot be known for certain that a vulnerability can be exploitable. So, these measures may be more misleading compared to the success rates that we determined through our work.

Second, some prior approaches use attack graph approaches that, given a set of vulnerabilities, determine the possible paths that attackers could follow until they reach their final goal [21–23]. These approaches use different approaches to choose, for a given vulnerability, which of the possible paths would be best or easiest to follow. We experiment on single step attacks and make the valid assumption that once a system is compromised, an attacker can attack the next system from there through horizontal or vertical privilege escalation. Using our results at each attack step could help determine the best path to follow. Our results could therefore be useful to these approaches and could provide a complimentary likelihood metric to assist in deciding the best direction to follow in an attack graph.

The third category of approaches deals with determining exploit metrics [24–26]. These approaches classify and characterise different forms of exploits. The approaches characterise the exploit binaries [24] or the exploit behaviour through traffic patterns [25]. In our work, we do not look at the detailed construction of exploits or the traffic patterns associated with their behaviour, but identify the existence of exploits and determine how likely they could be exploited on an identified target. There is no guarantee that the exploit metrics determined solely from binary characteristics and identified traffic patterns can be entirely meaningful, when we show in this work that not all exploits work as expected all the time.

The fourth and final category is the course of action (COA) metrics and measures. COA metrics provide supporting economic, security, or efficiency justification [27] in taking action to remedy a computer security problem. While all the numerical security measures determined through methods described above can be used for COAs support, some researchers [23, 28, 29] have put more emphasis on the COA as the main objectives of their work. Although they recommend a COA based on the approach's objective, it is generally left to the user to implement the best COA based on the metrics provided. These approaches could also benefit from the practical experimental success rates results obtained in this work.

CVSS [15], from the first category, is an open framework for sharing computer network security vulnerabilities and their potential impacts on enterprise networks. It is divided into three major metric groups, namely

Base, *Temporal* and *Environmental*. In the *Base* group of metrics is the calculated exploitability score that reflects on the likelihood of a vulnerability being exploited. It is based on the CVSS scores of access complexity, access vector, and authentication. However, this value, which provides a security metric for a single exploitation of a vulnerability, may not always be available or accurate [30]. In our work we used an experimental approach that determines this metric, which could be useful when the CVSS value is not available to supplement it or provide a more accurate measure if an estimate had been previously used.

5.0 DISCISSION AND CONCLUSIONS

Under the attacker model of an attacker that uses MSF, with only its library of exploits, and attacks server-side vulnerabilities, we observe that the Windows OSs only had eight matching exploits in the MSF library. Of those, six of them were exploitable on at least one of the OSs tested. Breaking it down further, two OSs had two exploitable vulnerabilities (in our attacker model), two had no exploitable vulnerabilities, and one had four exploitable vulnerabilities. This information shows that an organization must truly understand the type of attackers against whom they are defending. If they wanted to only defend against the basic attackers in our attacker model, there would be only six MS OS vulnerabilities to consider. Moreover, depending upon the particular OSs used by the organization, there could even be no vulnerabilities that are exploitable by that class of attacker.

Our data plainly demonstrates that the first launch of an attack often has a much higher success rate than later attacks do. When systems were rebooted after each attack, all of the exploitable vulnerabilities were exploitable 100% of the time. However, when the system was not rebooted, the subsequent attacks could have vulnerability success rates as low as 13%. In other words, after the first attack, the system is often in a state where subsequent attacks on the same vulnerability will fail. This indicates that penetration testing a vulnerable system can reveal whether that system was attacked in the past. If the system has an exploitable vulnerability that is exploitable 100% of the time on the first try, then if a penetration test fails to exploit the vulnerability, the conclusion is that the system was exploited before the penetration test by another attack. That means that the system could be compromised and should be treated in that manner.

The CVSS exploitability subscore gives an indication of how exploitable a vulnerability is, but our experiments show that the subscore is too coarse to be used as a metric for prioritizing mitigations on specific systems. A single exploitability subscore is given for all OSs. Our data shows that one OS could be reliably exploited 100% of the time by our attacker model while another OS (indicated by CVSS to be vulnerable and exploitable) could not be exploited at all by our attacker model.

Our work shows that vulnerabilities differ greatly in their exploitability. Metrics on the exploitability success likelihood for exploits in widely and freely available tools are required so that decision makers can efficiently manage their cyber security resources. If resources were unlimited, and if mitigating security measures caused no adverse side-effects, then an organization could mitigate every vulnerability that is disclosed. However, the reality is that human and financial resources to secure networks are often scarce, and mitigations can have significant usability and functionality side-effects. Our recommendation is for organizations to consider the types of attackers against whom they need to defend, and include the vulnerability-exploitability attributes as an input to their mitigation prioritization process. It is unrealistic for many organizations to have the ability to produce the necessary data themselves, so we recommend an information sharing approach, either through commercial or community ventures.

ACRONYMS AND ABBREVIATIONS

CVSS Common Vulnerability Scoring System

COA course of action

HTTP hypertext transfer protocol

IIS Internet Information Services

LSASS Local Security Authority Subsystem Service

MSF Metasploit Framework

MS Microsoft

MSDN Microsoft Developer Network

OS Operating System

OSVDB Open Source Vulnerability Database

R2 Revision 2

RPC remote procedure call

SP service pack

VM virtual machine

NVD National Vulnerability Database

REFERENCES

[1] O'Gorman, J., Kearns, D., and Aharoni, M., *Metasploit: the penetration tester's guide*, No Starch Press, 2011.

[2] Mateski, M., Trevino, C. M., Veitch, C. K., Michalski, J., Harris, J. M., Maruoka, S., and Frye, J., "Cyber threat metrics," Tech. Rep. SAND2012-2427, Sandia National Laboratories, March 2012.

[3] Bodeau, D., Graubart, R., and Fabius-Greene, J., "Improving Cyber Security and Mission Assurance Via Cyber Preparedness (Cyber Prep) Levels," *SocialCom 2010: IEEE Second International Conference on Social Computing*, Aug 2010, pp. 1147–1152.

[4] Parker, T., Sachs, M., Shaw, E., and Stroz, E., *Cyber adversary characterization: Auditing the hacker mind*, Syngress, 2004.

[5] Symantec Corporation, "Internet Security Threat Report 2014," *Symantec Corporation*, Vol. 19, April 2014.

[6] Rapid7, "Metasploit: The Attacker's Playbook," http://www.rapid7.com/products/metasploit/, 2014, Access date (2 April 2014).

[7] Offensive Security Ltd., "Metasploit Unleashed," http://www.offensive-security.com/metasploit-unleashed/Main_Page, 2014, Access date (2 April 2014).

[8] Core Security, "Core Impact," http://www.coresecurity.com/, 2013, Access date (15 October 2013).

[9] Net Marketshare, "Desktop Operating System Market Share 2012," http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0&qpsp=2012&qpnp=1&qptimeframe=Y, Access date (25 June 2013).

[10] NIST, "National Vulnerability Database," http://nvd.nist.gov, Access date (15 October 2013).

[11] OSVDB, "The Open Source Vulnerability Database," http://www.osvdb.org/, 2013, Access date (15 October 2013).

[12] Microsoft, "Microsoft Security Bulletins," http://technet.microsoft.com/en-us/security/bulletin, 2013, Access date (15 October 2013).

[13] Offensive Security, "Client side attacks," http://www.offensive-security.com/metasploit-unleashed/Client_Side_Attacks, 2014, Access date (22 January 2014).

[14] CORE Security, "CORE Security client-side exploits," http://www.coresecurity.com/core-security-client-side-exploits, 2013, Access date (22 January 2014).

[15] Mell, P., Scarfone, K., and Romanosky, S., "Common Vulnerability Scoring System," *IEEE Security Privacy*, Vol. 4, No. 6, Nov 2006, pp. 85 –89.

[16] Verizon, "2012 Data Breach Investigations Report," *Verizon*, 2012, pp. 1–92.

[17] Symantec, "Symantec Intelligence Report: September 2012," *Symantec*, 2012.

[18] tenable Network Security, "Nessus Vulnerability Scanner," http://www.tenable.com/products/nessus, Access date (15 October 2013).

[19] Houmb, S. and Franqueira, V., "Estimating ToE Risk Level Using CVSS," *ARES 2009: International Conference on Availability, Reliability and Security*, March 2009, pp. 718 –725.

[20] Joh, H. and Malaiya, Y., "A Framework for Software Security Risk Evaluation using the Vulnerability Lifecycle and CVSS Metrics," *Proc. International Workshop on Risk and Trust in Extended Enterprises*, 2010, pp. 430–434.

[21] Pamula, J., Jajodia, S., Ammann, P., and Swarup, V., "A weakest-adversary security metric for network configuration security analysis," *QoP 2006: Proceedings of the 2nd ACM workshop on Quality of protection*, 2006, pp. 31–38.

[22] Homer, J., Ou, X., and Schmidt, D., "A sound and practical approach to quantifying security risk in enterprise networks," *Kansas State University Technical Report*, 2009, pp. 1–15.

[23] Sawilla, R. and Skillicorn, D., "Partial Cuts in Attack Graphs for Cost Effective Network Defense," *HST 12: 2012 IEEE International Conference on Technologies for Homeland Security*, 2012, pp. 291–297.

[24] Langweg, H., *Software Security Metrics for Malware Resilience*, Ph.D. thesis, Sekretariat für Forschungsberichte, Inst. für Informatik III, 2008.

[25] Kolbitsch, C., Comparetti, P., Kruegel, C., Kirda, E., Zhou, X., and Wang, X., "Effective and efficient malware detection at the end host," *Proceedings of USENIX Security Symposium*, Montreal, QC, Canada, 2009.

[26] Rieck, K., Holz, T., Willems, C., and Patrick Dussel, a. P. L., "Learning and Classification of Malware Behavior," *Detection of Intrusions and Malware, and Vulnerability Assessment*, edited by D. Zamboni, Vol. 5137 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2008, pp. 108–125.

[27] Henderson, G., Sawilla, R., Matwin, S., Bacic, E., Tremblay, L., Yayyad-Shirabad, J., and de Souza, E. N., "Automated risk management system: Decision making support for continuous improvement of IT mission assurance," Technical Report TR 2012-060, DRDC-Ottawa, Ottawa, ON, Can, 2012.

[28] Homer, J. and Ou, X., "SAT-solving approaches to context-aware enterprise network security management," *IEEE Journal on Selected Areas in Communications*, Vol. 27, No. 3, april 2009, pp. 315–322.

[29] Sawilla, R. and Ou, X., "Identifying critical attack assets in dependency attack graphs." Tech. Rep. TM 2008-180, DRDC, 2008.

[30] Bhatt, S., Horne, W., and Rao, P., "On Computing Enterprise IT Risk Metrics," *HP Laboratories*, Vol. 1, No. HPL-2011-26, 2011.