A Look at Linux Kernel Vulnerabilities and their Effects from the Eyes of an Attacker

[1]Prasanna Sambasivan, [1]Syed Shariyar Murtaza, [1]Abdelwahab Hamou-Lhadj, [2]Mario Couture

[1]Software Behaviour Analysis (SBA) Research Lab
Concordia University, Montreal, QC, Canada
[2]System of Systems Section, Software Analysis and Robustness Group,
Defence Research and Development Canada, Valcartier, Québec, QC, Canada
[1]{p_sambas, smurtaza, abdelw}@ece.concordia.ca, [2]mario.couture@drdc.gc.ca

Abstract. Protecting the Linux kernel from malicious activities that could subvert its operations is of paramount importance. Different approaches have been proposed to analyze kernel-level vulnerabilities. The objective is to gain better understanding of the source and impact of attacks, which in turn can help in building appropriate mitigation mechanisms. Existing studies, however, are either outdated or have a strong focus on the type of attacks that may occur (e.g. buffer overflow) instead of analyzing the vulnerability from the attacker's perspective. In this paper, we report on our analysis of 301 Linux kernel vulnerabilities from 2009-2011. We classify these vulnerabilities from the attacker's view using various criteria such as the objective of the attacker when exploiting a given vulnerability, the targeted subsystem of the kernel, the location from which vulnerabilities can be exploited (i.e., locally or remotely), the impact of the attack on confidentiality, integrity, and availability of the system, and the complexity level associated with exploiting vulnerabilities. Our analysis of the 301 Linux vulnerabilities indicates the presence of a large number of low-complexity vulnerabilities. Most of them can be easily exploited from the local system (i.e., no need for remote access), leading to attacks that can severely compromise the kernel quality of service and allow attackers to gain privileged access.

Keywords: Software security, Linux kernel vulnerability, vulnerability and attacks taxonomies.

1. Introduction

Attacks on operating system kernels can cause serious damages to the entire host. The kernel consists of a large amount of code essential for the proper operation of various interconnected subsystems of the operating system. User applications interact with kernel subsystems typically through system calls, network connections, and I/O control mechanisms. Design faults in the kernel lead to vulnerabilities, which can be exploited maliciously by attackers, either locally or remotely, undermining the overall security and stability of user applications or even worse allowing access to unauthorized data. Techniques that ensure kernel security are therefore needed.

Recently, we started a research project on developing advanced host-based anomaly detection techniques. The project is a collaboration between Concordia University and the Defence Research and Development Canada (DRDC), a division of the Canadian Department of National Defence. It encompasses various research threads with a particular emphasis on protecting Linux-based platforms.

One of the key aspects of host-based anomaly detection techniques is the ability to decide whether a system is functioning properly or not. To this end, it is important to study what constitutes a normal or healthy behaviour of the system. The common approach is to measure various characteristics of the system in a lab environment that can later be used as a baseline for future comparisons. A fault detection technique can then be developed by observing, using monitoring capabilities, any deviations of the deployed system from these measurements [19].

Building effective host-based detection systems require good understanding of the host system to select aspects of the system that need acute monitoring. Monitoring all aspects of the Linux kernel, for example, would turn to be ineffective because of the large amount of collected information and the high overhead imposed by most monitoring techniques. Since our focus is on Linux-based platforms, we begun our research by studying the possible ways by which the Linux kernel can be compromised. The review of the literature (as we will describe in the related work section) showed that existing studies had either a broader scope by classifying attacks independently from the platform or a strong focus on the type of attacks (e.g., buffer overflow) caused by faults in the kernel. Many of them are also outdated. Although these studies have been useful in understanding the overall security threats the kernel is subjected to, they did not provide us with sufficient (and more practical) insight that we expected. For example, it was impossible to know which Linux kernel subsystems have been recently the target of most attacks. This information is important since it would allow us to target these components instead of the whole system. We were also interested in a number of other questions including:

- What is the cause and effect relationship between vulnerabilities in the kernel and the attack objectives? In other words, we want to understand which vulnerabilities attackers have exploited to cause a desired objective such as obtaining sensitive information or causing a denial of service.
- How many attacks have a partial or full impact on CIA (confidentiality, integrity and availability) attributes?
- How many vulnerabilities of the kernel can be exploited locally vs. remotely?
- What is the complexity level required to exploit a given vulnerability?

We believe that the answer to these questions enables security experts to have a better understanding of Linux kernel vulnerabilities and their effects, and hence build better defence mechanisms that fit their needs. To achieve our objective, we analyzed 301 Linux kernel (ver. 2.6.x) vulnerabilities documented between 2009 and 2011 on the National Vulnerability Database (NVD) [1]. In this paper, we share our findings along with our recommendations.

The rest of this paper is structured as follows: In Section 2, we report on related work. In Section 3, we describe the dataset used in our study. In Section 4, we present the criteria by which we classified the attacks along with the analysis of the attacks. Section 5 focuses on the threats to validity. We conclude our work in Section 6.

2. Related Work

There is a large body of research that aims to classify attacks with different focuses that vary from general-purpose computing systems to internet applications, wireless networks, etc. (e.g. [2, 3, 9, 7, 12]). In this section, we only report on attacks that are relevant to Unix-like systems.

Bishop presents a general taxonomy of Unix system and network daemon vulnerabilities [8]. The goal of their paper is to describe vulnerabilities in a format useful to intrusion detection and prevention techniques. Further, the paper discusses methods for finding these vulnerabilities and preventing exploits of these vulnerabilities.

Chen et al. [4] classify information from 141 previously documented kernel vulnerabilities, and then analyze how current runtime prevention techniques (e.g., software fault isolation, code integrity checking, user mode drivers, and memory tagging/tracking) address the prevention of these vulnerabilities. The authors also present insights on the usage of compile time static code analysis tools to detect bugs in the kernel.

Mokhov et al. [5] introduce taxonomy of methods to mitigate vulnerabilities in the kernel. They have examined 290 documented vulnerabilities from 2002-2007 in the kernel and classified them by the type of error (e.g., design, input validation, buffer overflow etc.). Different categories are established on the basis of how the vulnerabilities were patched. Some of these resulting categories include changing the data type, precondition validation before execution, zeroing memory before use, input validation and fail safe default initialization. These categories are then further combined with current programming guidelines to form security-oriented programming guidelines for the Linux kernel.

Argyroudis et al. [17] analyze the current countermeasures built into operating system kernels to prevent common kernel exploits such as NULL pointer dereferences. Subsequently, the paper overviews various memory corruption mitigation techniques and proactive mitigations used by various operating systems (Linux, Windows, Mac OS X, iOS and Android). Techniques that are used to bypass such kernel protection mechanisms are also briefly discussed.

The main difference of our work with prior studies is that existing approaches revolve around techniques that can prevent the occurrence of vulnerabilities and their exploits instead of describing the vulnerabilities and their characteristics from the attacker's perspective, which is the main focus of this paper.

3. The Attack Dataset: National Vulnerability Database (NVD)

The Linux kernel vulnerability dataset used in our study comes from NVD, which is a US Government repository of vulnerability information [1] used a reference site in the area of security. It contains a set of documented vulnerabilities covering a wide large of software systems. Each vulnerability is recorded using the following attributes:

- A unique ID known as the Common Vulnerabilities and Exposures (CVE) ID.
- A short description that contains information about the affected software, the attack method, the cause of the vulnerability, and the objective that can be achieved upon successful exploitation.
- A vulnerability score calculated using a standardized scoring mechanism that we will explain in the next paragraphs.
- The potential impact on confidentiality, integrity and availability if the vulnerability is exploited.
- The complexity level for accessing and exploiting the vulnerability.

Table 1. Example of a reported vulnerability in NVD

CVE ID	CVE-2010-1146
Description	The Linux kernel 2.6.33.2 and earlier, when a ReiserFS filesystem exists, does not restrict read or write access to the .reiserfs_priv directory, which allows local users to gain privileges by modifying (1) extended attributes or (2) ACLs, as demonstrated by deleting a file under .reiserfs_priv/xattrs/.
Cvss Score	6.9
Confidentiality Impact	Complete
Integrity Impact	Complete
Availability Impact	Complete
Access Complexity	Medium

Table 1 shows an example of a vulnerability entry in NVD. In this example, the vulnerability appears in the Linux kernel ver. 2.6.33.2 and earlier versions. The vulnerability, if exploited, can allow local users to gain privileged access. It can also impact confidentiality, integrity, and availability of the system. Exploiting this vulnerability is of medium complexity.

The vulnerability score is calculated using the Common Vulnerability Scoring Mechanism (CVSS) [6], which is an open vulnerability framework used to assign a score to a vulnerability by taking into account unique characteristics of the given vulnerability such as the impact and complexity of the attack. The higher the score, the more dangerous the vulnerability is.

A search query for “Linux kernel” on the NVD yields approximately 850 vulnerability records from Linux kernel 2.2 since 1999. A total of 301 vulnerabilities affecting the Linux kernel version 2.6.x have been documented in the NVD from January 2009 – November 2011. We used these are the vulnerability dataset used in this study. We only considered version 2.6.x of the kernel to allow proper interpretation of the results without a loss of generality. The breakdown of the vulnerabilities per year is reported in Table 2.

Table 2. Year-wise breakdown of vulnerability dataset

2009	99
2010	125
2011 (until November)	77
Total	**301**

4. Classification of Linux Kernel Vulnerabilities

Table 3 shows the criteria by which we classified the Linux kernel vulnerability dataset. We selected these criteria to provide good understanding of the effect of vulnerabilities from different angles. We elaborate on each criterion and present the results of classifying the Linux kernel vulnerabilities in the subsequent sections.

A. Attack Objective

Through the examination of NVD vulnerability records for the Linux kernel 2.6.x, we found that the objectives of the attackers can be grouped into six categories: (1) making resources unavailable, (2) allowing access to confidential system information, (3) bypassing access restriction mechanisms, (4) obtaining elevated privileges (5) executing random code, and (6) spoofing identity. It should be noted however that some exploited vulnerabilities may have lead to combined effects such as gaining privileges and bypassing security restrictions.

Table 3. Criteria by which we classified vulnerabilities

Category	Description
Objective of Attack	What effect the attack has on the Linux kernel
Affected Component	The component of the kernel that is vulnerable.
Origin of Attack	Locally exploitable, local network exploitable, remote network(Internet) exploitable
Access Complexity	Need of privileges, special conditions or other vulnerabilities.
Impact	Complete
Impact on CIA	Impact on confidentiality, integrity and availability.

Figure 1 shows the results of classifying the Linux kernel vulnerabilities on the basis of the attack objectives. Vulnerabilities for which we could not find information regarding the attack objective in the NVD repository are listed as unspecified.

A close examination of the results in Figure 1 shows that a high number of vulnerabilities (153/301) lead to denial of service (DoS). These attacks alone do not cause loss of integrity or breach of system confidentiality. DoS vulnerabilities, however, should not be regarded as low risk since the combination of DoS with other attacks can compromise the entire Linux kernel as noted by Chen et al. [4]. The authors showed how three different vulnerabilities including DoS vulnerability are used together to completely compromise the Linux kernel.

Figure 1 shows that vulnerabilities that allow attackers to obtain sensitive information come in the second position with a total of 55 out of 301 vulnerabilities, followed by vulnerabilities that cause bypassing security restrictions and gaining privileged access (such as root-level access).

We found that a relatively small number of vulnerabilities provide attackers with the ability to spoof identity or execute arbitrary code. We suspect that this is due to the continuous improvements made to the Linux kernel to prevent such attacks from taking place.

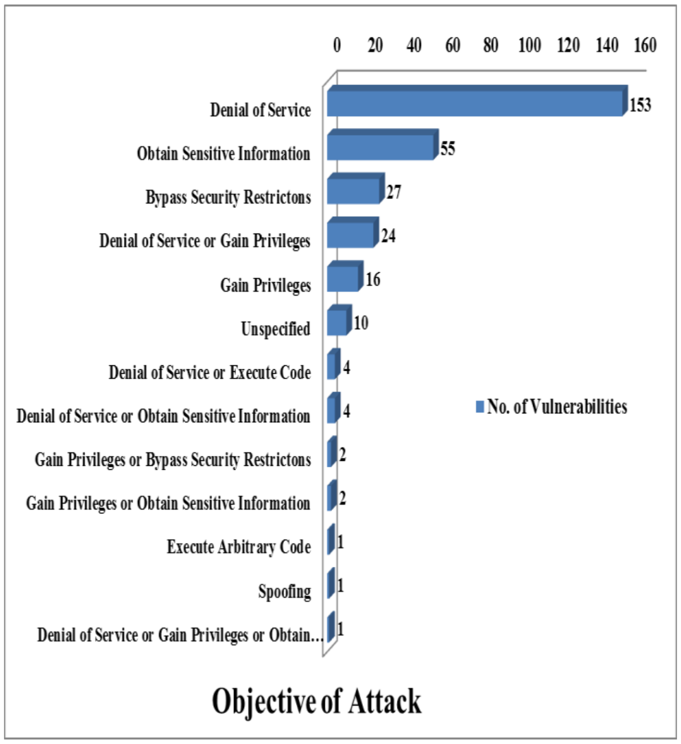


Figure 1. Attack distribution by objective

B. Affected Component

This category denotes the Linux kernel components targeted by the attackers. By component, we mean a subsystem of the Linux kernel (e.g., arch, net, fs, crypto, etc.). The main Linux kernel components are shown in Table 4 (for more details on these components, please refer to [14]). Attacks exploit vulnerabilities found in specific functions within the same component of the kernel.

Table 4. Components of the Linux kernel

arch/	fs/	lib/	security/
block/	include/	mm/	sound/
crypto/	init/	net/	tools/
drivers/	ipc/	samples/	usr/
firmware/	kernel/	scripts/	virt/

In Figure 2, we show the distribution of the attack per kernel component. For some vulnerabilities, the NVD lacks information on the affected functions. These represent 10 out of 301, which we classify as unspecified in Figure 2.

It can be observed from Figure 2 that the *fs and net* alone account for 50% of the total vulnerabilities. This is worrisome since these components are vital to the functioning of the OS. The drivers, on the other hand, account for 20% of the total vulnerabilities. Not all drivers are however needed during operation. In addition, military systems use usually a very small set of drivers. It is therefore hard to assess the teu impact of driver vulnerabilities on a system in operation. But the threat still exists.

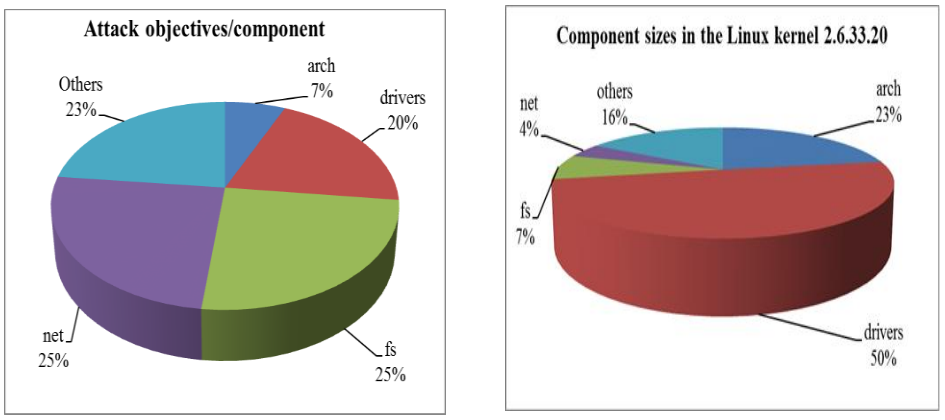


Figure 2. Distribution of attacks across components

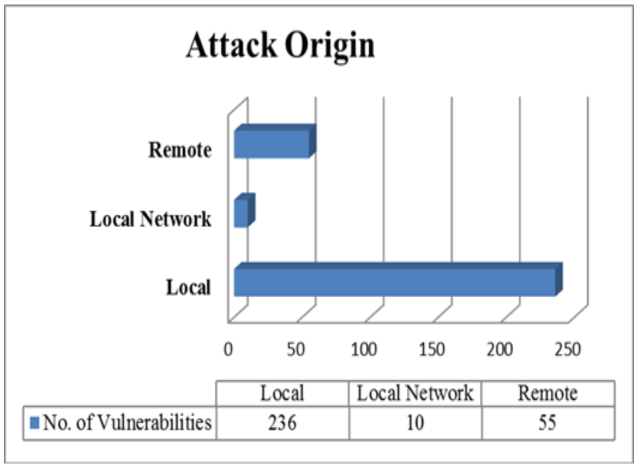
We correlated the number of vulnerabilities with the size of the kernel components. Figure 2 (right pie) shows the distribution of the size of the kernel components. One interesting observation is that although the *net* and *fs* components accounts for only 11% of the size of the kernel, they constitute 50% of the total vulnerabilities. On the other hand, the *drivers* component, which accounts for 50% of the kernel size, constitutes only 65/301 (20%) vulnerabilities. This suggests that thorough testing is needed for the vital components of the system to uncover vulnerabilities. Also, host-based anomaly detection systems that operate at the kernel level should focus attention on monitoring and observing the behaviour of these components and ensure that deviations from normalcy are caught.

C. Attack Origin

There are three ways by which a vulnerability can be exploited:

- Locally from within the system
- From the collision domain/broadcast domain of the target system network
- From a remote location such as public Internet

It should be noted that a vulnerability that is exploitable from a remote location is also exploitable from the other two network locations, whereas the reverse is not valid. Attackers would need to originate their exploits from one of these three location points. Studying the distribution of kernel vulnerabilities across the three locations would allow security experts to examine whether host based prevention systems or network based prevention systems must be given more priority to prevent the occurrence of these exploits. The CVSS scoring mechanism includes the *origin of attack* as one of the scoring parameters. We used this indicator to categorize the location by which a vulnerability can be exploited.



	Local	Local Network	Remote
■ No. of Vulnerabilities	236	10	55

Figure 5. Vulnerability distribution by origin of attack

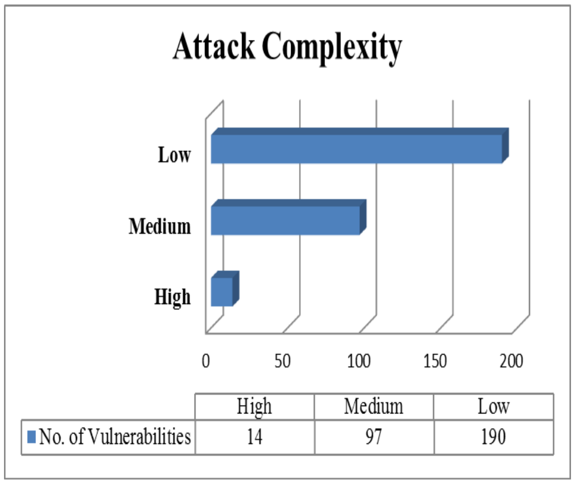
It can be observed from Figure 5 that approximately two-third (66%) of the total number of vulnerabilities (236/301) are locally exploitable, which means that exploitation would require some sort of local access to the system. An important distinction must be made between remotely exploitable vulnerabilities and remote access to a system to aid in understanding the difference between local and remote vulnerabilities. While a remotely exploitable vulnerability can be exploited directly over a network connection, remote access to a system simply refers to accessing the system by some remote mechanism such as *ssh* or *telnet*. We classified the latter as local exploits.

The high prevalence of locally exploitable vulnerabilities suggests that host-based intrusion prevention techniques should be given attention since event the most sophisticated network based intrusion prevention systems would serve little purpose in the prevention of such local attacks.

D. Attack Complexity

The level of expertise of the attackers and target system requirements to exploit vulnerabilities vary from vulnerability to another one. Some vulnerabilities may, for example, require some services to run on the target system, specific system architecture, the presence of a particular device driver, etc. The CVSS scoring mechanism defines three complexity levels: low, medium, or high to each vulnerability. Low complexity means that the exploitation is trivial and can be performed using readily available scripts. Medium rating means that some pre-defined conditions must be met. High complexity is used when the vulnerability is exploitable only in a specific environment, or when specific conditions (such as elevated privileges or the presence of additional vulnerable components) are met.

Figure 6 shows kernel vulnerabilities with respect to the complexity of attacks. Examination of these results shows that over 60% (190/301) of vulnerabilities are low-complexity exploits.



	High	Medium	Low
■ No. of Vulnerabilities	14	97	190

Figure 6. Vulnerability distribution by attack complexity

We examined the attack complexity level based on the attack objective. As shown in Table 6, we see that many attacks to gain privileges, obtain sensitive information, or bypass security restrictions, are of low to medium complexity.

Additionally, we also determine the vulnerability distribution obtained by comparing access complexity against the origin of attack in Table 5. We found that about 50% (146/301) of the vulnerabilities of low complexity are exploitable locally, which enforces the need for advanced host-based anomaly detection techniques.

Table 5. Attack objective versus attack complexity

Attack Objective	Low	Medium	High
Denial of Service	98	46	9
Obtain Sensitive Information	30	24	1
Bypass Security Restrictons	20	7	
Denial of Service or Gain Privileges	14	9	1
Gain Privileges	9	6	1
Unspecified	7	2	1
Denial of Service or Execute Code	4		
Denial of Service or Obtain Sensitive Information	3	1	
Denial of Service or Bypass Security Restrictons	1		
Denial of Service or Gain Privileges or Obtain Sensitive Information	1		
Execute Arbitrary Code	1		
Gain Privileges or Bypass Security Restrictons	1		1
Gain Privileges or Obtain Sensitive Information	1	1	
Spoofing		1	
Total	190	97	14

In addition, the fact that most kernel vulnerabilities discovered are easily exploitable hint that developers and security experts must place even more emphasis on security during kernel design to help proactively address and uncover these vulnerabilities before they are found by others.

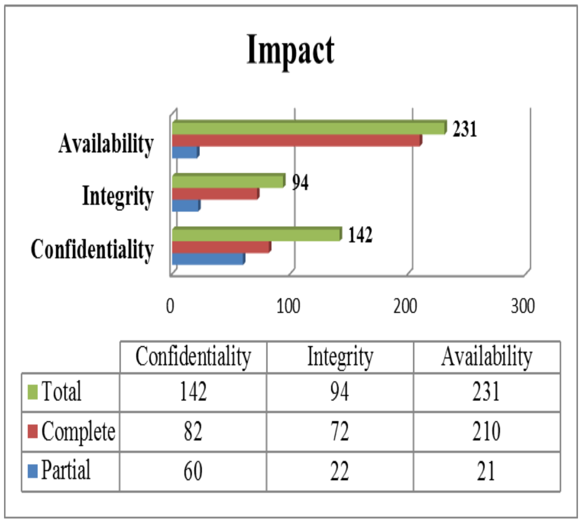
Table 6. Origin of attack versus attack complexity

Network Access required	Access Complexity			Total
	Low	Medium	High	
Local	146	78	12	236
Local Network	5	4	1	10
Remote	39	15	1	55
Grand Total	190	97	14	301

E. Attack Impact on CIA

In this category, we classify the impact of vulnerabilities on CIA (confidentiality, integrity, availability) attributes. Confidentiality refers to restricting information to authorized users. Integrity entails ensuring that the information is presented as intended by the owner. Availability refers to the ability of the system to provide services.

The distribution of attacks by the impact they have on confidentiality, integrity and availability is reflected in Figure 7 below. As expected, the presence of a large number of vulnerabilities that causes denial of service result in attacks that hinders the availability of the system.



	Confidentiality	Integrity	Availability
🟩 Total	142	94	231
🟥 Complete	82	72	210
🟦 Partial	60	22	21

Figure 7. Vulnerability distribution by impact.

We also found that only a small number of vulnerabilities cause *partial* impact; i.e., the impact of these vulnerabilities cannot be controlled by the attacker. In most cases, the impact is *complete,* i.e., a potential attacker can control what components of the kernel can be compromised, hence increasing the likelihood of a targeted attack.

Additionally, a high number of vulnerabilities that cause *complete* confidentiality and integrity impact. Thus, attackers can choose from a larger set of vulnerabilities to launch targeted attacks that negatively impact *at least* one of the quality factors of Linux kernel architecture.

Table 9 shows the relationship between the attack impact and the origin of the attack. As we can see, a significant number of vulnerabilities that impact confidentiality (130/142), integrity (85/94) and availability (172/231) are locally exploitable. The ratio of locally exploitable vulnerabilities that cause *complete* impact to the total number of vulnerabilities is high (73/82, 65/72, and 152/210 for confidentiality, integrity and availability respectively).

From the above statistics, it is evident that in most cases, attackers can control the impact that vulnerabilities will have on the kernel. Since the impact can be controlled, targeted attacks towards a specific function can result, implying that the quality of service offered by the kernel is likely to be severely affected. Therefore, there is a need to have mechanisms that would limit the impact of a successful attack on the kernel, if not prevent them. Such mechanisms may require minor modifications in kernel architecture.

Table 7. Impact versus origin of attack

		Local	**Local Network**	**Remote**	**Grand Total**
Confidentiality Impact	Complete	73	2	7	**82**
	Partial	57	1	2	**60**
	Total	**130**	**3**	**9**	**142**
Integrity Impact	Complete	65	2	5	**72**
	Partial	20	0	2	**22**
	Total	**85**	**2**	**7**	**94**
Availability Impact	Complete	152	9	49	**210**
	Partial	20	0	1	**21**
	Total	**172**	**9**	**50**	**231**

F. Discussion

In this paper, we analyzed 301 vulnerabilities of the Linux kernel from 2009 to 2011 reported on the NVD. Our findings along with recommendations are:

1. Denial of service (153/301), obtaining sensitive information (55/301), bypassing security restrictions (27/301) and gaining elevated privileges (16/301) are the main objectives of attack. This suggests that techniques that ensure high-availability of services of Linux-based systems are warranted to counter the effect of DoS.
2. The *arch* (20/301), *net* (75/301), *fs* (75/301) and *drivers* (65/301) subsystems of the kernel are most frequently reported vulnerable and hence the likely target of exploits. These components require special attention from developers and security testers. Also, host-based detection techniques should put an emphasis on acute monitoring of these components.
3. Majority of the vulnerabilities are locally exploitable (236/301). Thus host-based intrusion detection systems should be given attention in conjunction with network-based intrusion detection systems for attack prevention.
4. Exploiting vulnerabilities in 190 out 301 cases is of low to medium complexity, making it easy for attackers with basic skills to attack the kernel. More emphasis must thus be placed on security during kernel design, to proactively address such vulnerabilities. Additionally, most of these low complexity attacks are locally exploitable, further emphasizing the importance of host-based intrusion detection systems.
5. A large number of vulnerabilities have *complete* impact on confidentiality (82/142), integrity (72/94) and availability (210/231). Most of these vulnerabilities that have

complete impact can be locally exploited, which further justifies the need for host-based detection techniques.

5. Threats to Validity

In this section, we outline the threats to the validity of our analysis in the form of four categories: construct validity, conclusion validity, internal validity and external validity [18].

A threat to construct validity exists due to the choice of categories in our classification. Since this paper looks at Linux kernel vulnerabilities from an attacker's perspective, the categories of our taxonomy are all attack-centric (objective of attack, affected component, origin of attack, complexity of attack and impact of attack). These five categories most adequately reflect how attackers would use vulnerabilities to cause exploits. Other categories that classify vulnerabilities from the attacker's perspective may exist, but we think that the ones presented in this study provide good coverage on the way vulnerabilities could be exploited, which can help in designing effective defence mechanisms.

The conclusion validity is threatened because of the inherent assumption in this paper that every vulnerability corresponds to a possible attack via an exploit. Though, there exists a theoretical possibility that attacks exist for every such vulnerability reported on the NVD, the actual exploits might not exist for many of these vulnerabilities, or exploits might not be possible. This threat is mitigated by the fact that a majority of the exploits have possible attacks.

A threat to internal validity arises due to the lack of available information about vulnerabilities. Some vulnerabilities lack information about the attack objective or the affected component. In such cases, we have manually added entries based on our understanding of these vulnerability descriptions, which if incorrect, may have skewed statistics of our empirical analysis. However, such cases have been extremely rare and do not affect the overall conclusions. Additionally, categories that have directly been used from the CVSS scoring mechanism are exhaustive, error free and adequately informative across all the 301 vulnerabilities.

The number of representative vulnerabilities that have been selected threatens the external validity. A total of 301 vulnerabilities over three years have been analyzed. While this forms a large representative subset of the total number of vulnerabilities in the 2.6.x version of the kernel, the numbers and percentages of vulnerabilities in every category may have minor distortions as vulnerabilities from earlier years and earlier versions of the Linux kernel are accounted for.

6. Conclusion

In this paper, we presented our analysis of 301 vulnerabilities of the Linux kernel ver. 2.6.x from 2009 to 2011. We classified these vulnerabilities using various criteria that describe the way an attacker can exploit a given vulnerability.

Our analysis shows that the kernel has a large number of low-complexity vulnerabilities that can be easily exploited from the local system (i.e., without the need for remote access location). A large number of vulnerabilities can be exploited to cause denial of service. Vulnerabilities that permit bypassing security restrictions and gaining elevated privileges are also an important threat.

Our future work is to continue the study of the Linux kernel and its components (especially the ones that are the target of most attacks) in order to build better monitoring techniques that can detect deviations from a prior model that presumably captures the normal behaviour of the kernel.

Acknowledgment

This work is partly supported by the NSERC (Natural Sciences and Engineering Research Council of Canada) and DRDC (Defence R&D Canada), Valcartier, QC, Canada.

References

[1] National Vulnerability Database URL: http://nvd.nist.gov

[2] C. Simmons, C. Ellis, S. Shiva, D. Dasgupta, Q.Wu. “AVOIDIT: A Cyber Attack Taxonomy,” *Technical Report: CS-09-003, University of Memphis,* 2009.

[3] D.Lough. ‘A Taxonomy of Computer Attacks with Applications to Wireless Networks’, *PhD Thesis Dissertation, Virginia Polytechnic Institute and State University*, 2001.

[4] H. Chen, Y. Mao, X. Wang, D. Zhou, N. Zeldovich, M. Frans Kaashoek, “Linux kernel vulnerabilities: State-of-the-art defenses and open problems,” *In Proc. of the Second Asia-Pacific Workshop on Systems,* 2011.

[5] S. Mokhov, M. Laverdière, D. Benredjem, “Taxonomy of Linux Kernel Vulnerability Solutions,” *In Proc. of the International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering,* 2007.

[6] P. Mell, K. Scarfone, S. Romanosky, “A Complete Guide to the Common Vulnerability Scoring System version 2.0,” URL: http://www.first.org/cvss/cvss-guide.html.

[7] G. Alvarez and S. Petrovic, “A new taxonomy of web attacks suitable for efficient encoding,” *Elsevier Journal on Computers and Security, 22(5):435–449, 2003*.

[8] M. Bishop, “A taxonomy of Unix and network security vulnerabilities,” *Technical report, Department of Computer Science, University of California at Davis*, 1995.

[9] J. D. Howard, “An analysis of security incidents on the Internet,” *PhD thesis dissertation, Carnegie Mellon University, Department of Engineering and Public Policy*, 1997.

[10] J. D. Howard and T. A. Longstaff, "A common language for computer security incidents," *Technical Report SAND 988667, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California*, 1998.
[11] S. Kumar, "Classification and Detection of Computer Intrusions," *PhD thesis dissertation, Purdue University*, August 1995.
[12] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi, "A taxonomy of computer program security flaws, with examples," *ACM Computing Surveys, 26(3):211–254*, 1994.
[13] U. Lindqvist and E. Jonsson, "How to systematically classify computer security intrusions," *IEEE Symposium on Security and Privacy*, pp. 154–163, 1997.
[14] Linux Cross Reference. http://lxr.free-electrons.com
[15] N. Elhage, "CVE-2010-4258 - Turning denial-of-service into privilege escalation," URL: http://blog.nelhage.com/2010/12/cve-2010-4258-from-dos-to-privesc/
[16] A. Tripathi, U. Singh, "Analyzing Trends in Vulnerability Classes across CVSS Metrics," URL: http://research.ijcaonline.org/volume36/number3/pxc3976282.pdf
[17] P. Argyroudis, D. Glynos, "Protecting the Core Kernel Exploitation Mitigations," *Black Hat Europe*, 2011.
[18] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, A. Wesslen. *Experimentation in Software Engineering: An Introduction*. Spring, 2002.
[19] D. Ghosh, R. Sharman, H. R. Rao, S. Upadhyaya , "Self-healing systems -survey and synthesis," *In the Journal of Decision Support Systems, 42(4),* 2007.