NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEVELOPMENT TIME OF ZERO-DAY CYBER EXPLOITS IN SUPPORT OF OFFENSIVE CYBER OPERATIONS

by

Konstantinos Bompos

September 2020

Thesis Advisor: Alan B. Shaffer
Co-Advisor: Gurminder Singh

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE DEVELOPMENT TIME OF ZERO-DAY CYBER EXPLOITS IN SUPPORT OF OFFENSIVE CYBER OPERATIONS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Konstantinos Bompos	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) OPNAV N8	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Zero-day vulnerabilities are those that have not previously been identified and thus are in their zeroth day of existence. These vulnerabilities are the most potentially damaging from a cyber defense perspective because the defender is unaware of their existence and a malicious attacker can exploit them to take control of a system without the owner's consent or knowledge. Zero-day vulnerabilities are also highly valuable to offensive cyber operations as they may be exploited before defenders are aware of their existence or can patch their systems to adequately defend them. This comprehensive study of zero-day vulnerabilities is focused on showing them to be a vital factor in cyber operations. Offensive cyber operators can benefit from techniques that help accelerate the development time of a zero-day exploit. In contrast, defenders and vendors can reduce their response time by improving their methodology to discover and patch zero-day vulnerabilities. This research provides an extensive review of zero-day vulnerabilities and examines their overall impact on targeted system security. We present characteristics of a system that increase its susceptibility to zero-day vulnerabilities and security measures to improve the zero-day vulnerability awareness of the defender. We also propose techniques for reducing the development time of zero-day exploits to enhance offensive cyber operations.

14. SUBJECT TERMS cyber, security, cybersecurity, exploit, zero-day vulnerability, zero-day exploit, offensive cyberspace operations	**15. NUMBER OF PAGES** 77
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

DEVELOPMENT TIME OF ZERO-DAY CYBER EXPLOITS IN SUPPORT OF OFFENSIVE CYBER OPERATIONS

Konstantinos Bompos
Captain, Hellenic Army
BS, Hellenic Army Academy, 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2020

Approved by: Alan B. Shaffer
Advisor

Gurminder Singh
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Zero-day vulnerabilities are those that have not previously been identified and thus are in their zeroth day of existence. These vulnerabilities are the most potentially damaging from a cyber defense perspective because the defender is unaware of their existence and a malicious attacker can exploit them to take control of a system without the owner's consent or knowledge. Zero-day vulnerabilities are also highly valuable to offensive cyber operations as they may be exploited before defenders are aware of their existence or can patch their systems to adequately defend them. This comprehensive study of zero-day vulnerabilities is focused on showing them to be a vital factor in cyber operations. Offensive cyber operators can benefit from techniques that help accelerate the development time of a zero-day exploit. In contrast, defenders and vendors can reduce their response time by improving their methodology to discover and patch zero-day vulnerabilities. This research provides an extensive review of zero-day vulnerabilities and examines their overall impact on targeted system security. We present characteristics of a system that increase its susceptibility to zero-day vulnerabilities and security measures to improve the zero-day vulnerability awareness of the defender. We also propose techniques for reducing the development time of zero-day exploits to enhance offensive cyber operations.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

APT	advanced persistent threat
ASLR	Address Space Layout Randomization
BYOD	bring your own device
CO	Cyberspace Operation
CP	contingency plan
CTF	Capture the Flag
CVE	Common Vulnerabilities and Exposures
DCO	Defensive Cyberspace Operations
DLLs	dynamic link libraries
DMZ	Demilitarized Zone
DOD	Department of Defense
DODIN	Department of Defense Information Network
HIPS	Host Intrusion Prevention System
IoT	Internet of Things
IPC	inter-process communication
IS	information systems
ISO	International Organization for Standardization
ITSEC	Information Technology Security Evaluation Criteria
JP 3-12	Joint Publication 3-12
LIFO	last-in-first-out
MitM	Man-in-the-Middle
NOPs	No-operation instructions
OCO	Offensive Cyberspace Operations
PoC	Proof of Concept
SCADA	Supervisory Control and Data Acquisition
SMBv1	Server Message Block version 1
SP	security policy
TCSEC	Trusted Computer System Evaluation Criteria

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Professor Shaffer and Professor Singh, for their guidance and mentorship over the last several months. I would also like to thank all the members of the International Graduate Programs Office and my program officer, LCDR Eric Regnier, for their support during my attendance at the Naval Postgraduate School.

Most importantly, I would like to express my very profound gratitude to my beloved wife, Lina, and my little daughter for providing me with unfailing support and continuous encouragement throughout my life. It would not be the same without you. You make my life complete.

Finally, I devote the completion of this thesis to my beloved father, Andreas, who passed away in August 2019. His loss, while it had an emotional impact on me, also inspired and gave me the courage to have my thesis completed.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

A vulnerability in computer software is a loophole in the computer code and/or its configuration that can possibly compromise the security of a computer system [1]. An adversary can exploit such vulnerabilities and take control of a system without the approval or, in many cases, without the knowledge of its user. Among an adversary's actions, zero-day exploits make use of a vulnerability whose existence is still unknown to computer system defenders and software vendors. These vulnerabilities are potentially the most damaging from a cyber defense perspective because the defender has no knowledge of their existence, and a patch for the affected software has not yet been developed [2].

The security level of a computer system is often measured by the total count of vulnerabilities on the system and the time required to exploit them, or the time required for the implementation and distribution of software patches to mitigate the vulnerabilities. In the face of zero-day vulnerabilities, these measures can become skewed because a full understanding of existing vulnerabilities is not possible. For the adversary, developing a reliable zero-day exploit can be challenging, but it can be extremely rewarding since it will be unknown to defenders, and ideally, will work on any configuration of a vulnerable target [3].

Zero-days attacks are more frequent today than they were in the past, and they remain undercover for long periods of time, making them much more pernicious than other types of attacks. The time to develop a zero-day exploit must include the time to identify a zero-day vulnerability, and then the time to conduct reconnaissance against target systems. This development time will also depend greatly on the skill and risk tolerance of the attacker, as well as the security configuration of defended systems. Research in 2009 showed that 80 percent of critical vulnerabilities had a functional exploit within ten days of their public disclosure [4]. Moreover, RAND Corporation in 2017 reported that, once the vulnerability researchers detect an exploitable vulnerability, the time for developing a reliable exploit is relatively short, with a median time of 22 days [3].

For these reasons, this comprehensive study of zero-day vulnerabilities is focused on showing them to be a vital factor in cyber operations. Offensive cyber operators could benefit from techniques which help accelerate the development time of a zero-day exploit. In contrast, defenders and vendors could reduce their response time by improving their methodology to discover and patch zero-day vulnerabilities.

B. RESEARCH OBJECTIVES

The objective of this work is to examine the development time of zero-day exploits and how it can affect cybersecurity in current enterprise systems. The research also examines techniques for reducing the development time of these exploits in order to support offensive cyber operations.

C. RESEARCH QUESTIONS

We achieved the objectives of this thesis by focusing on the following research questions:

1. Primary Question

How does the existence of zero-day vulnerabilities in computer systems, and the time for adversaries to develop exploits against these vulnerabilities, impact the overall level of cybersecurity of that system?

2. Secondary Question

How might the total time to develop zero-day exploits, including the time to identify associated zero-day vulnerabilities, be reduced to better support offensive cyber operators?

D. BENEFITS OF STUDY

There are two main contributions of this research. The first is to survey and draw together the field of disparate resources about zero-day vulnerabilities into one study. The second contribution benefits the area of offensive cyber operations by providing a thorough study of useful techniques for reducing the development time of zero-day exploits. By more

widely disseminating information and knowledge about this topic, the report enables those who design security systems to create systems that are even more impervious to malicious actors.

E. THESIS ORGANIZATION

The remainder of this thesis is organized into the following chapters.

1. Chapter II: Background

Chapter II provides background research to understand the basic terminology concerning zero-day vulnerabilities and zero-day exploits and attacks. We analyze the zero-day vulnerability cycle and its impact on cybersecurity, and the importance of zero-day vulnerabilities in offensive and defensive cyber operations (OCO/DCO).

2. Chapter III: Information System Security and Zero-Day Vulnerabilities

Chapter III examines the dynamic concept of information security. It analyzes the procedures for system configuration and their contribution to defense against zero-day exploits.

3. Chapter IV: Zero-Day Attacks

Chapter IV describes the impacts of zero-day vulnerabilities to cyber system defense. It analyzes potential characteristics that can make a system more susceptible to zero-day vulnerabilities, and suggests countermeasures and guidelines for increasing zero-day vulnerability awareness. Finally, it examines the life cycle of exploit development, and how knowledge of a zero-day vulnerability can affect it beneficially.

4. Chapter V: Conclusion and Future Work

Chapter V summarizes the research results of this thesis. Additionally, the chapter provides recommendations for future work to further extend this research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

Software vulnerabilities are a significant cause of cybersecurity problems. Zero-day vulnerabilities represent a unique category, as they remain unknown to defenders, and they are prone to exploitation before software vendors release patches. Those who have knowledge of vulnerabilities may create exploits and perform cyber-attacks to take control of a computer. This chapter provides an overview of the fundamental concepts and types of vulnerabilities. The focus of this work will be zero-day vulnerabilities and their importance in cyber operations.

A. VULNERABILITIES OVERVIEW

Vulnerabilities in computer security are considered as weaknesses in a system that can grant access to a malicious user and provide unwilling any kind of advantage in that system. The International Organization for Standardization's (ISO) standard on information security risk management defines a "vulnerability as a weakness of an asset or group of assets that can be exploited by one or more threats" [5]. Taken together, a vulnerability and a threat that may exploit that vulnerability constitute a risk to the system.

Vulnerability exploitation can be performed through the use of specific tools, depending on the vulnerability present on the system, with varied results. Thus, a vulnerability could be considered a steppingstone, possibly leading to a system compromise. However, this does not necessarily mean that all vulnerabilities lead to security risks to the system.

For example, a vulnerable system that contains no sensitive data may be prone to exploitation, but compromising it might not be considered a security risk since the malicious user would gain nothing of value from it. Such an example is a honeypot, which is a purposely created vulnerable system containing no critical data or services. Honeypots are used to either identify and trap potential attackers, or to stall their progress by leading them to deadends on a network.

There are huge collections of ever-expanding databases consisting of known vulnerabilities that are used to help test and identify potential weaknesses in systems. One of the most prominent of these is the Common Vulnerabilities and Exposures (CVE) system. CVE provides a list of publicly known vulnerabilities, where each entry corresponds to an identification entry number, a brief description of the vulnerability, and at least one public reference [6].

1. Importance and Applications of Vulnerabilities

In terms of importance, the existence of security vulnerabilities is the cornerstone of computer hacking and exploitation. Vulnerability exploitation is the way most, if not all, systems are compromised, and it is a matter of fact that vulnerabilities will never cease to exist. One way or another, most systems prove to be vulnerable to something eventually, such as flaws in coding, system setup, or even human error. Security vulnerabilities sometimes even transcend computer systems since attempts like social engineering could be performed without any technical skills being put in use. A company or an organization that is not sufficiently prepared by educating its employees on these matters is considered vulnerable in this case. In this manner, the existence of security vulnerabilities is one of the most critical aspects of information security. Ensuring secure systems is a constant act of discovering and fixing new vulnerabilities, while at the same time, maintaining those systems in order to avoid being affected by already discovered vulnerabilities.

According to Erickson [7], vulnerabilities can be categorized as design, implementation, and operational. Design vulnerabilities are those that are caused by mistakes or oversights in the way software has been designed to operate. In this category of vulnerabilities, computer software operates exactly as it was intended to, but the problem lies in the fact that it was designed to do the wrong thing. Design flaws are also known as high-level vulnerabilities, or issues with program requirements or constraints. There is a distinction, however, in design flaws between software requirements and software specifications. Every software systems design is driven by requirements. That plainly means that every computer program has a specific purpose or scope, and it is designed to accomplish some set of objectives that fill that purpose. So, in a sense, requirements are

the list of objectives to be accomplished by the program. After software requirements have been specified, they are used by software engineers as guides to construct the plans for how the program should be constructed to meet those requirements. These plans are called the program's specifications [7].

In the case of implementation vulnerabilities, the source code is working as intended, i.e., it meets the design specifications, but the security problem lies in the way the program's operation are carried out. Problems such as this often occur due to technical discrepancies, when a solution that deviates from the design is implemented. Implementation vulnerabilities are also known as low-level or technical flaws. The practice of code reviewing immensely helps in the identification of implementation vulnerabilities since they are mostly caused by technical artifacts or similar nuances in the developed code. Some of the most common implementation vulnerabilities are buffer overflows and SQL injections, which will be explained later in Section 2.

Operational vulnerabilities cannot be identified by reviewing a program's source code because they occur during the interaction between program software and its environment. They are generally caused by issues with software configuration in a specific environment, or by automated and manual processes supporting the system. Operational vulnerabilities can be exploited by attacks mainly directed to users, like social engineering and theft. The distinction between operational and design or implementation vulnerabilities is not entirely clear. This is mostly due to the fact that there is overlap in these concepts, so these distinctions are mainly to be used as references.

2. Types of Computer System Vulnerabilities

Following is list of some commonly known vulnerability types:

a. Bugs

When the result from a software or a system is not accurate or unexpected is generally due to what is known as a software bug. A software bug exists in a system or a program as an error or a flaw and can make it to behave in unintended ways [8]. Various techniques or tools are used to pinpoint bugs, a process most notably known as

"debugging." Additionally, most of the modern computer systems are embedded with proper software that, during their operation, is responsible for preventing, detecting, and auto-correcting several software bugs to protect the system from halting.

Errors and mistakes in the general design, in the source code, or in the components that are used by a program could be the cause of several bugs. Incorrect code produced by compilers may also cause some bugs. A program is called defective when it consists of too many or too severe bugs that compromise its intended functionality. However, the effects of bugs could be subtle or catastrophic by triggering errors with ripple effects that finally will crash or halt the execution of the system.

Some bugs qualify as security bugs. Security bugs are especially harmful because they could potentially enable a malicious user to gain control of the vulnerable system by manipulating the data passed and handled by it in order to perform in unintended ways.

b. Weak passwords

Passwords are the most common authentication tool used in security to ensure that sensitive data does not get compromised [9]. An effective password is determined by its strength and its ease to be remembered by the user. A strong password means that it is crafted in a way that cannot be guessed easily. In that sense, weak passwords are those that contain easily accessed knowledge about a user (birthdays, names), or easy character sequences that can be found in common wordlists. An insecure password can be compromised by a brute force attack, a type of attack that utilizes computing power using trial-and-error to guess the targeted password.

Brute force attacks can be either blind, trying combinations of different characters, or targeted by using a wordlist. A wordlist is generally a text file that includes commonly known passwords (the most common passwords being sequences of "a" or "1," or strings such as "qwerty" or "12345…"), and is used by the brute force tool potentially speeding up the process in case the password tested is included in the file. These types of brute force attacks are commonly known as dictionary attacks. Some of the most notable brute force attack tools are JohnTheRipper and Hydra, but many other variations target specific platforms as well. Most password cracking systems consist of some brute-force attack

protection techniques such as measuring the intervals between each password input, meaning that too fast betrays an automated attack taking place, or by limiting the times of failed attempts to a certain number, negating trial-and-error attempts as a result.

As previously mentioned, a password is effective when it is easy to use and can be easily remembered by the user. A generally bad practice is having a potentially strong password written down on physical or digital notes, thereby nullifying its strength if it falls on the wrong hands. This risk is mitigated by password management software such as KeePass [10], which helps maintain the integrity of many different strong passwords for the user.

c. Virus-infected software

A virus is a type of code attached to software that is generally being executed along with the original code, writing and replicating itself into a target system [11]. The virus code uses elevated privileges in order to replicate itself and can execute various commands totally unintended by the original program, resulting in data breach or destruction. Virus code writers use critical system knowledge along with social engineering techniques to spread the virus to the target system or network of systems. A virus can be masked by the extension of a file, for example, or by having its code injected on the source code of a program or a file, masking itself from the user. Most viruses are also crafted in ways to avoid detection from anti-virus software altogether. Executing the legitimate program causes the virus to be unintentionally executed as well.

The majority of viruses target Microsoft Windows operating systems [12]. This is partly because of its popularity, but also because these operating systems share similar or identical footholds to be exploited, meaning that a virus targeting a Windows distribution is effective in all similar distributions. Operating system diversity protects against viruses in this way, since a virus crafted for a Windows distribution cannot operate in a Linux environment. This does not mean that Linux operating systems are impervious to computer viruses, which is a common misconception.

By default, Linux and Unix operating systems prevent normal users from tampering with the operating system environment without permission, but this is not generally true

for Windows operating systems, leaving them vulnerable to viruses in case of user error [13]. A virus called “Bliss” was created in 1997 targeting Linux distributions. A requirement for Bliss to execute was that it be run by the user, thus it would have the same privileges on files of that user. As a result, it was unlikely for the virus to tamper with the operating system since most users in Linux distributions do not have administrative or “root” access alternatively. The Bliss virus was created mostly for research purposes and never really became widespread.

d. Lack of data encryption

A severe vulnerability that might occur is the absence of encryption on sensitive data prior to its storage or transmission [7]. This vulnerability makes Man-in-the-Middle (MitM) attacks possible, where a malicious user attempts to highjack transmissions between the vulnerable system and another party, leading to sensitive data compromise. MitM attacks are generally prevented by encrypting data prior to transmission, thus making it much more difficult for an eavesdropper to process without the decryption key.

e. OS command injection

OS command injection is a security vulnerability, also known as shell injection [14]. It allows the arbitrary execution of OS commands, which usually used by a malicious user to elevate his privileges in the vulnerable system. This action can easily result in data loss or compromise since the malicious user could potentially gain access to all of the target system’s resources. OS command injection can be executed through vulnerable web services or poorly designed software. To mitigate this vulnerability, it is highly recommended to sanitize user inputs to system and application software since this is the only way a user could inject OS commands.

f. SQL injection

SQL injection is another kind of command injection vulnerability [15]. Instead of executing operating system commands, a malicious user performing SQL injection attempts to execute commands on the database management system running of a vulnerable system. The results of a successful SQL injection could vary from stolen data to massive

data loss. In the worst-case scenario, the whole backend database system could be compromised. SQL injection is possible when user input is not correctly sanitized, thereby allowing a malicious user to pass unintended malicious commands to the database.

g. Buffer overflow

Buffer overflow vulnerabilities occur when user input data exceeds what the program was initially designed to handle [16]. This excess data is written to nearby memory blocks, corrupting other data, or overwriting it with malicious code. As a result, the affected program does not behave as it was intended, and this irregular behavior could be manipulated to affect the vulnerable system in various ways.

Susceptibility to buffer overflow vulnerabilities varies depending on the programming language used. Some older programming languages such as C or C++ are more prone to buffer overflow exploitation since they require the programmer to handle memory allocation manually. Languages like PHP or Python are less susceptible to buffer overflows because they automate the memory handling process. This, however, does not mean that they are completely invulnerable. Finding a buffer overflow vulnerability can be quite difficult, however, potential buffer overflow vulnerabilities consist of serious risks and may lead to data compromise, denial of service, or even a complete system takeover through privilege escalation.

Two are the primary type of buffer overflow vulnerabilities, the first one is the *stack overflow*, while the second one is the *heap overflow*. Stack buffer overflows consist of buffer overflows on the *stack*, which is the memory space that the operating system uses to store function return addresses, and local variables. Data storage or retrieval on the stack is organized in the last-in-first-out order (LIFO). On the other hand, heap buffer overflows occur by manipulating the heap, or the memory space used as dynamic data storage. Heap memory reservation and management is controlled by an executing program instead of the operating system.

h. Ineffective access control

The access control software is responsible for performing the authorization checks every time an actor is trying to access a resource or proceed to execute an action. [17]. In the case of a user with valid credentials, authorization is the method of deciding if the user will need to access a specific resource, taking into consideration the user's context rights or permissions or different access control requirement which have been provided to the resource.

In the cases that effective access control checked are not enforced or implemented, users would have access to information or perform operations that under different access control checks would not be permitted [17]. This may result in improper data leakages, denial of service, and arbitrary code execution.

i. Use of broken algorithms

The use of weak or flawed algorithms, or the incorrect application of stable algorithms, might lead to sensitive information been leaked [18]. A determined attacker can easily identify a non-standard algorithm and break it, and this action will compromise whatever data have been protected. Coding errors in design, implementation, or programming logic increase the probability an attacker already knows or could gain the knowledge and the tools to subvert the security mechanisms.

j. Missing authentication for critical function

This type of vulnerability is a flaw in the code of a vulnerable application that occurs when there is a lack of proper authentication before performing essential tasks that might cause damage to the system [19]. As an example, requiring no mail authentication before changing the password to an account could lead to the account being taken over by an attacker quite easily.

k. Unrestricted upload of dangerous file types

Unrestricted user file upload could prove to be a critical system vulnerability [20]. A malicious user could attempt to exploit this vulnerability by uploading dangerous file

types like executables that could severely affect or compromise the system. Uploading and executing a reverse shell, for example, is a common way to get unauthorized access to a system. The consequences of this exploitation depend on what the malicious software is programmed to do with the uploaded files, and additional attention should be paid where the files will be stored. Files upload exploitation with no restrictions could vary in results; some of them could be that the complete system would be taken over, overloaded the system, client-side attacks, redirecting the attacks to a back-end system, or just a system defacement.

This vulnerability also can be applicable to web server applications, where it is known as command injection vulnerability [21]. These web applications are used for the convenience of the user when he needs to interact with the underlying operating system, the file system, or a database. However, if the user's inputs are insufficiently validated, severe security problems may occur. In this way, the host operating system could execute arbitrary operating system commands which are issued by an attacker. The term command injection attacks are referring to attacks that occur once an application or user accepts unvalidated user data to the operating system shell, such as HTTP headers, form, cookies, etc. [22]. In that event, very careful validation of input data should take place when a web application is allowed to pass user inputs to web server functions.

l. Dependence on untrusted inputs in a security decision

An untrusted user can modify inputs to an application in ways that bypass the protection mechanism of that application when these mechanisms rely on the basis that the inputs are trustworthy [23]. It is a common mistake for programmers to consider that trusted user data inputs such as hidden form fields, or cookies are impossible to modify. A malicious user could manipulate these values to gain an advantage over the system. This action is undetectable, and as a result, security decisions taken by these inputs may lead to attacks that getting around the security of the operating system or software. Any input that comes from an outsider to a system should be considered trustworthy by default.

m. Download of code without integrity checks

When an application proceeds to download source code or executable files from untrustworthy locations without performing any verification on the code, then the integrity of the system that contains this application is at risk [24]. Malicious users could have dangerous code being executed by compromising the host server, modification of the code on transit, or DNS spoofing.

n. Race conditions

Race conditions describe vulnerabilities that result from the timing of actions that affect other types of actions on a system [25]. A race condition can occur under a variety of situations, from tracing processes to filesystem accesses, but it is especially prevalent in multithreaded and distributed software programs [26].

Any system that involves a multiprocessing environment is vulnerable to race condition attacks. Such an attack can happen when multi-threaded applications are executed and they differ in the sequence in which they are being executed [27]. They generally involve one or more applications using the same resource. Until the resource becomes available, the multiple threads might use a key, called a semaphore, to control the multiple access and define which process can use the resource. If the semaphore is used improperly, or if the resource crashes, all processes will be waiting indefinitely, while no process is using the resource [27]. Race condition vulnerabilities receive limited attention, as they often go unspotted and it is difficult to eliminate them completely.

B. ZERO-DAY

In this section, we analyze the concept of a zero-day, to include zero-day vulnerabilities and what differentiates them from other common vulnerabilities.

1. Meaning and Basic Concepts

Zero-day is a term that bears some misconception since it has been widely misused, especially by the media. The earliest known use of the term was in 1998 in an e-zine called CRH, which described “Our member chameleon set us up with a domain in Argentina, the

D-Lab... It has some mad [stuff] on it, but you have to know where to look, because www.d-lab.com.ar will take you nowhere, it has 0-day exploits on it, as well as other useful stuff and source code, check it out" [25]. The term was first commonly used to describe pirated versions of a legitimate product being made available to the public at the same time or before the official release.

Nowadays, though, in terms of information security, zero-days describe something entirely different [28]. Vulnerabilities that are tagged as zero-day refer to their being unknown by those who must mitigate them. Until a viable solution is provided to mitigate a zero-day vulnerability, attackers can exploit it, resulting in programs, data, other computers, or entire networks being compromised. A zero-day, in this case, refers to the fact that the deadline in days for developers to mend the vulnerability is zero, meaning that their systems are probably already exposed to attack threats.

Examining the perspectives of attackers and defenders regarding zero-day vulnerabilities, to attackers a zero-day vulnerability is a highly valuable way of accessing a target system [2]. Users and system administrators regard those types of vulnerabilities as a serious security risk for their clients. The same applies to vulnerable software vendors, since these vulnerabilities pose a serious threat to their clients and by extension to their own integrity and reputation.

2. Zero-day Vulnerabilities: A Unique Category

Based on research findings on zero-day vulnerabilities, there are some key facts that distinguish them from vulnerabilities in general. Zero-day vulnerabilities are not publicly disclosed, although sometimes software vendors know about bugs in their software and flaws that have not yet been patched.

A FireEye report states that vulnerabilities identified by adversaries could stay publicly unknown on average of 310 days [29]. According to Symantec's Internet Security Threat Report, in the year 2015, a new unknown vulnerability was discovered every seven days. Also, during the last few years, zero-day vulnerabilities are notably increasing, as 23 zero-day vulnerabilities were reported in 2013, 24 in 2014, and 82 in 2016 [30].

Another interesting fact is that zero-day vulnerabilities are mostly used in targeted attacks with only a few hosts as targets; the exceptions being a few high-profile attacks like Stuxnet [2]. Public disclosures of zero-day vulnerabilities seem to be followed by a massive increase of attacks that are taking advantage of any insufficient patch or the zero-day exploit that was used. According to research statistics, it is assumed that exploits of around 42% of the attacks that were based on an exploitable zero-day vulnerability stay active within 30 days after they are publicly disclosed [2]. For cyber criminals, zero-day vulnerabilities in well-approved software like Microsoft Office and Adobe Reader serve as a free access point that allows them to compromise any target they want. Consequently, the market value of a new unpatched vulnerability ranges between $5000-$250,000 [31]. Based on these numbers, it is safe to assume that cybercriminals are constantly on watch for disclosure of new vulnerabilities so that they can exploit them.

3. Life cycle of a Zero-Day Vulnerability

Software vendors try to identify software vulnerabilities early in order to patch them and fix bugs with periodic software updates. Unfortunately, sometimes vulnerabilities are discovered by other actors, and exposed to exploitation before they can be patched by the vendor. In this situation, the software programmers had “zero days” to patch the vulnerabilities before they were exploitable, and were disclosed to the public. Thus, they became known as *zero-day vulnerabilities*.

The life cycle of a zero-day vulnerability consists of seven phases, as shown in Figure 1. A zero-day vulnerability typically starts as a software bug. If the bug remains undetected by the software vendor, attackers might be able to exploit it, performing what it is called a *zero-day attack* against targets that exhibit the vulnerability. An attack can be characterized as a zero-day when vulnerabilities not yet publicly disclosed are exploited in order to perform the attack. When a vulnerability is discovered by software vendors, its information is usually disseminated to the public as a security issue. After that, the vendors release patch(es) for the zero-day vulnerability, and update anti-virus signatures with the new zero-day attack information.

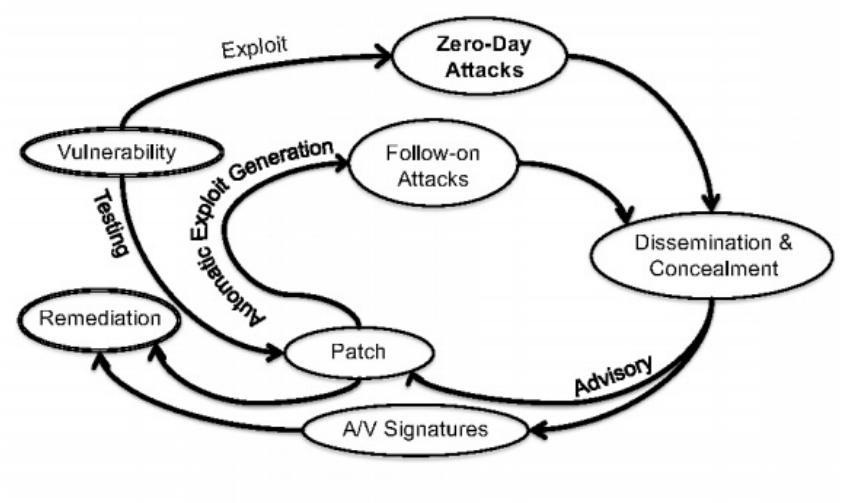

Figure 1. Zero-day Vulnerability Life Cycle. Source: [2].

After a vulnerability has been discovered and patched by vendors, new exploits might still be developed and employed against targets that have not applied the patch yet. This cycle of patching-and-exploiting can take several years before remediation comes, and the vulnerability eventually ceases to affect systems.

The life cycle of a zero-day vulnerability can also be marked in a timeline as shown in Figure 2.

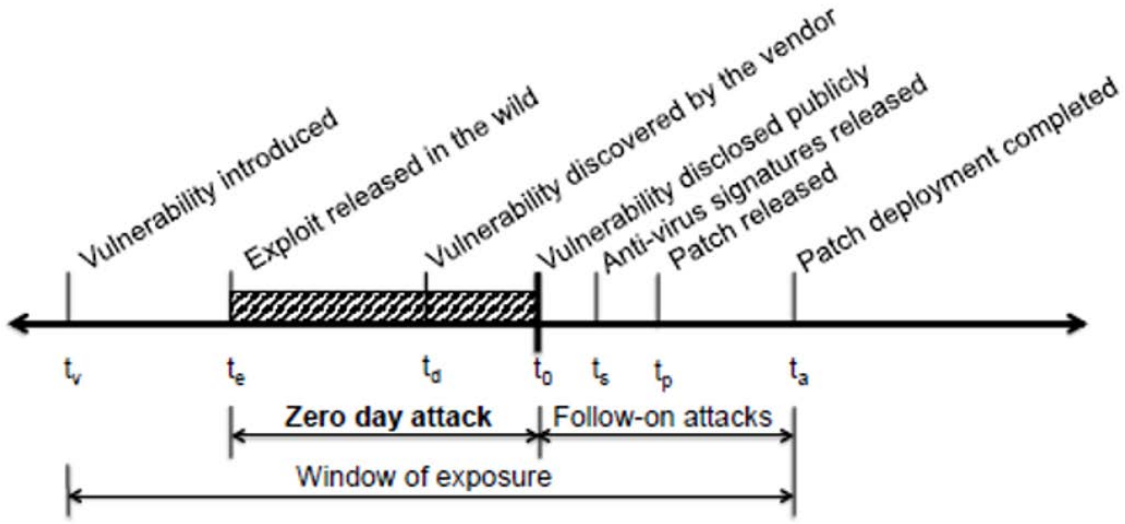

Figure 2. Attack Timeline. Source: [2].

- **Vulnerability introduced (t_v).** This is the first phase where a bug is introduced to software that is being used by hosts worldwide, thus creating the vulnerability.

- **Exploit released in the wild (t_e).** Vulnerability researchers identified that the vulnerability is exploitable. A fully functional exploit is created and used to compromise the security of targeted hosts. It is during this phase when exploits using zero-day vulnerabilities are being sold to interested parties.

- **Vulnerability discovered by the vendor (t_d).** Through the process of internal testing or by a third-party report, the vulnerability is eventually discovered. As a result, the issue is exposed to the software vendor.

- **Vulnerability is disclosed publicly (t_0).** At this point in time, the vulnerability is announced to the public either by the vendor, or by forums, blogs or mailing lists, along with a CVE identifier assigned to it.

- **Anti-virus signatures released (t_s).** Following vulnerability disclosure to the public, anti-virus vendors update their software by updating the signature database for the ongoing attacks and eventually protecting the end-host that now can detect them from the new anti-virus signature.

- **Patch released (t_p).** A patch fixing the zero-day vulnerability is released by the vendor. This action mitigates the risk of exploitation on hosts that apply the patch.

- **Patch deployment completed (t_a).** The vulnerability ceases to affect the end hosts after they apply the patch.

However, the events in the timeline in Figure 2 do not always appear in the given order, but in general, the relation is $t_a > t_p \geq t_d > t_v$ and $t_0 \geq t_d$. Additionally, the association between t_d and t_e cannot be identified generally. By these definitions, it should be clear that the "zone" for a zero-day attack to occur is in the period between t_e and t_0 [2].

4. Real-World Examples of Zero-day Attacks

It is a common misconception that zero-day attacks and zero-day exploits are the same thing. This is far from the truth, however, there is a direct correlation between the two terms. A zero-day exploit is targeting software flaws and is intending to access data, behave as another user, or even take control of a system without the approval or, in many cases, without their knowledge of the user [32]. A zero-day attack is the actual operation of a cyber-attack using the exploit code to compromise a specific target.

In February 2013, a zero-day attack was reported in Adobe Reader 10 and 11 that was able to bypass the contained sandbox anti-exploitation protection feature. Experts later concluded that the exploit used was in fact, so highly sophisticated that it was either developed by a nation-state as a cyber-espionage tool, or belonged to private contractors supporting law enforcement and used for communication interception purposes [33]. Another notable example of a zero-day attack was when Sony Pictures Entertainment fell victim to such an attack during 2014. The reasons behind the attack were political, and the aftermath left the company with approximately 100 terabytes of valuable data being stolen, including email addresses of senior executives, contracts, business plans, and user accounts, along with four unreleased feature films [34].

Microsoft almost fell victim to a zero-day attack on June 2016 when a zero-day exploit targeting Windows OS was found on sale in the dark web [34]. This exploit made possible a local privilege escalation on Windows 10 systems since it was able to grant any user account with administrative privileges. It is unknown, however if the exploit was sold eventually or not. The Democratic National Committee hack, on the other hand, is known as one of the most widely covered cyber-attacks in history. Attacks began when thousands of spear phishing e-mails were sent to target employees working for the DNC. E-mails contained malicious links to phishing pages, giving total control to the user's PC and its internal network to attackers. Six zero-day exploits were found to be utilized by attackers aiming at operating systems, Microsoft Windows 10 included, as well as Adobe Flash Player and Java [34].

Arguably, the most famous zero-day attack was the one known as Stuxnet, which was publicly disclosed in 2010, even though it is suspected that the actual attack took place earlier. Stuxnet is one of the earliest known digital weapons developed, employing a highly infectious, self-replicating worm that was developed by one or more unnamed nations [35]. Stuxnet's final goal was to reprogram an industrial control system and the attack was mainly designed to target and destroy Supervisory Control and Data Acquisition (SCADA) equipment being used in Iran's nuclear weapons program, which was being used in nuclear fuel enrichment processes. Stuxnet is the first incident that was spread to worldwide media and marked the beginning of a new era on cyber-warfare conduct [36]. Regarding its technical details, this was the first time an exploit used several zero-day vulnerabilities simultaneously, along with a variety of other vulnerabilities as well. That fact marks Stuxnet as a highly sophisticated cyber weapon clearly ahead of its time.

C. CYBER OPERATIONS

Cyberspace as a domain provides opportunities, challenges, and the advantage of allowing actions that may be taken anonymously across the entire globe. Cyber-attacks may occur whatever the location may be, without worrying about barriers like physical access or weather conditions [37]. The U.S. Department of Defense defines cyberspace as a "notional environment in which digitized information is communicated over computer networks" [38]. With that defined, cyberspace operations consist of those operations employed with the intention to achieve objectives through cyberspace. The activities of cyberspace operations are executed in three separate but synchronized mission areas: Offensive Cyberspace Operations (OCO), Defensive Cyberspace Operations (DCO), and Department of Defense Information Network (DODIN) Operations [39].

1. Offensive/Defensive Cyber Operations

In terms of cyber warfare, offense has a clear advantage over defense because it is immensely easier to conduct a successful cyber-attack compared to defending against one [40]. This is true because the ultimate goal of technical defense would be to avoid the endanger of a target, where it is assumed the defending party must be able to defend the system successfully against each and every attack, in contrast, the attacking party will need

to be successful only one time [41]. However, the reliance on cyberspace for every operation drives defenders to continuously try to find ways to preserve the confidentiality, availability, and integrity of their systems for any known and unknown threats or vulnerabilities. Defenders need to determine their security needs by often conducting a risk analysis for their organization. Given the facts, it is realistically impossible for a defender to eliminate all system risk entirely. They can only mitigate risks by following proactive cyberspace security actions and security best practices. The Internet was originally designed to establish reliable, and easy connections and security was not a priority back in those days. As a result, connection is established even if the true identities of the connector and the incoming connection are obfuscated. In this regard, zero-day vulnerabilities are especially prized by offenders since they offer a sure way to compromise a target system unknowingly to its defenders.

Regarding cyber operations, coordination between offensive and defensive teams of the same group must be maintained. The reason behind this is that a cyber-attack launched at a given target could result in a counter-response from the adversary party, directly threatening defense. Also, when launching a cyber-attack, disruption of crucial civilian activity is a probable scenario that results in defensive implications.

Steps were taken to ensure defense against cyber-attacks are mostly proactive, aiming to harden and monitor defending systems. However, the larger the environment to be defended the more difficult it is to implement those defenses. Even though monitoring on large scale is possible nowadays, more sophisticated activities like intrusion detection and the conduct of vulnerability assessments can only be performed successfully on a smaller scale. In general, a strong defense against cyber-attacks consists of forming and maintaining strong security policies along with effective surveillance and tracking communication moving in and out of the target system. Intrusion detection and prevention on a smaller scale are essential as well. The main tools on maintaining a solid defense are the regular conducts of vulnerability assessments and penetration tests. Precautions have to be taken into consideration as well, with disaster recovery planning in case an attack succeeds to some degree can help mitigate the damage. Implementation of defense in depth is also a successful approach in which security is layered with several different lines of

defense standing between the attackers and the data to be protected. All in all, the main goal of defense is to slow down the attackers long enough to detect them or even completely fend off their operations. However, maintaining impenetrable defense for an indefinite period of time is unrealistic, so in the worst-case scenario, measures have to be taken to ensure damage mitigation.

Whereas defense usually revolves around protecting a large group of interconnected systems, attacking is usually focused on a particular system or a small set of them. The steps of a successful attack are reconnaissance, further vulnerability scanning, accessing, privilege escalation, and finally exploitation. Reconnaissance focuses on information gathering on the target, in order to collect specific information that will support the attack procedure. The target is then scanned for potential vulnerabilities to be used against. Access to the system is then obtained through vulnerability exploitation, but in most cases, this level of access does not possess enough privileges to perform the attack. Privilege escalation is the process that gives attackers administrator privileges or root access, through which they can accomplish their goals. Throughout the whole attack process, activity obfuscation must be maintained for the attackers to avoid detection. This is usually accomplished by removing traces of activity on target, such as system logs.

2. Cyber-attack vs. Cyber Exploitation

While cyber exploitation and cyber-attack might easily be confused, though they share some similarities, they are quite different in legal constructs as well as in their objectives [42]. This misconception is largely the result of both using similar technology, as well as sharing similarities in operational considerations. The main thing that distinguishes a cyber exploitation from a cyber-attack is in the payload. In both types of cyber actions, a vulnerability must be found first and then exploited, however only the cyber-attack typically launches a payload. Mission objectives greatly differ in general, as the mission of cyber exploitation is to compromise the confidentiality of valuable data on the target, whereas a cyber-attack would be more likely to be directed at destroying data to harm the target. However, it is important to note that a payload can execute several functions at the same time, both for cyber-attack and cyber exploitation. Due to the strong

relationship between technologies used in both actions, the cost of acquisitioning a software or a tool for cyber exploitation with the capability to cyber-attack is likely to be low.

There are several case scenarios on what an adversary might aim to accomplish during a cyber exploitation. For example, the adversary could from just observing the attached network structure and the data traffic, be able to exploit information on that network. In this case, the cyber exploiter would aim to gather valuable information while being undetected in order to exploit the target when the time is right. Valuable information like plans and codes could be extracted that way from unsuspecting adversaries. Another course of action would be to gain confidential information through the organization's network, which could be beneficial for their competitors.

A cyber exploitation operation should have a little known signature, which will greatly aid the success of the operation since it will typically consist of many smaller individual actions that will be done throughout the course of the attack, the preparation for the attach it would even take weeks or months. The target may even be in a position to tell the difference between a cyber-attack and a cyber-exploitation, which is even more difficult to distinguish when the time frame between the operations is short. Legal authorities required for those operations are quite different from each other, so clarity of operations is essential to avoid any conflicts. As a result of sharing similar approaches, developing expertise at cyber-attacks mostly accomplishes developing knowledge for conducting cyber exploitation, and vice versa.

Coordination between cyber-attack and cyber exploitation teams on a group is essential. Without this, several issues could occur, for example, having the exploitation team retrieve confidential data, only to prove later that the cyber-attack team planted this same data to confuse the adversary party. Both teams must consult each other between operations since shutting down a target's communication channel, while probably causing disarray to the adversaries, would also prevent the extraction of valuable data.

D. CHAPTER SUMMARY

A *vulnerability* is a type of software bug considered to be a security risk that leads to a system weakness. Malicious code can be written to *exploit* or take advantage of system vulnerabilities, and seize the control of a machine not only without the user's permission but even without the user's knowledge. There are many different types of vulnerabilities that can result from software bugs, weak passwords, and virus-infected software.

Some key facts distinguish *zero-day* vulnerabilities from vulnerabilities in general. Zero-day vulnerabilities are those which are exposed to exploitation before they are publicly disclosed and patched by the vendor. Zero-day exploits are mainly an advantage for an adversary who is associated with zero-day vulnerabilities, and are used in prominent cyber-attacks against high profile targets like the ones against Microsoft by the Stuxnet worm, as well as in defensive cyber operations.

Among the many actions taken in cyberspace, cyber-attacks can be most advantageous, as the attacker need not worry about location barriers. Cyber-attacks and cyber exploitation share some operational similarities but have significant differences in terms of payload and mission objectives. Despite this fact, their coordination is essential for successful cyber operations.

In the next chapter, our center of attention is the dynamic concept of information security, a domain that is constantly evolving alongside the evolution of networks and computers. We will analyze the procedure of the system's configuration and the importance of a well-secured configuration in defending against zero-day exploits.

III. INFORMATION SYSTEMS SECURITY AND ZERO-DAY VULNERABILITIES

Information is the most critical asset for modern organizations. The process of information security is fundamental to minimizing security risks, responding at the same time to the rapid evolution of the information environment and computer systems. In this chapter, we focus on information systems security as it continues to develop as an organizational function. The configuration of a system is our first line for an effective defense against zero-day exploitation. We then discuss the principles that need to characterize a well-secured configuration, and lastly, we examine an actual cyber-attack to emphasize the severe impact of a zero-day attack in a poorly configured environment.

A. INFORMATION SYSTEMS

There are several definitions of information systems (IS) in the literature. Laudon and Laudon [43] define an IS as a set of related components in an organization that are responsible for the collection, processing, storage, and distribution of necessary information essential for decision-making. Also, this information assists managers in finding solutions for various problems and promotes coordination and control inside the organization. For our analysis, this definition is sufficiently detailed and comprehensive.

An IS contains information about the organization and the environment around it. As seen in Figure 3, an IS performs three key functions—input information, process them, and finally output—to create the data that the owner needs, while feedback will be required for the evaluation, and the improvement of the input. As previously mentioned, the environment around would also need to communicate with the internal organization and its information system. The environment around can differ depending on the organization from a customer to a supplier, or even both, potential competitors and stakeholders [43].

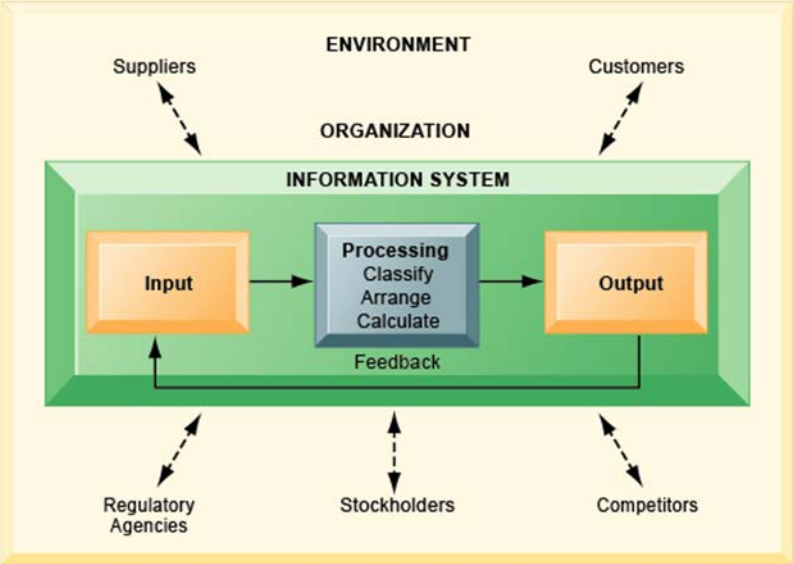

Figure 3. Functions of an Information System. Source: [43].

B. INFORMATION SYSTEMS SECURITY

IS security, often called InfoSec, refers to a set of procedures, human and technological resources, and processes that are responsible for the protection of an organization's information. Information worth being protected has some critical characteristics essential to the organization's operation. These include information confidentiality, integrity, and availability, called the "CIA Triad" in InfoSec. Some security practitioners also add accuracy, authenticity, utility, and possession to this triad to sufficiently respond to the continually evolving computer environment [44].

InfoSec is not a specific technology and cannot be bought off the shelf. It is a dynamic process that must be undertaken alongside the evolution of the organization's network itself. This process extends beyond technical issues to promote system safety and eliminate potential security incidents successfully.

Security is mostly about protecting assets from severe threats that may exploit system or network vulnerabilities. In the case of InfoSec, this definition is extended beyond system resources and refers to all forms of information itself. Cyberspace as a domain is contained in the information environment, and as a result, there is a distinct interdependency between cyber activities and InfoSec activities [45].

The DOD's Joint Publication 3-12 (JP 3-12) provides joint doctrine and military guidance on cyberspace operations. JP 3-12 outlines a three-layered representation of cyberspace to better support the planning and execution of cyberspace operations: physical network, logical network, and cyber-persona. While each of the layers has a different focus, they all are interrelated and somewhat reliant on each other [45].

1. Physical Network Layer

The physical layer consists of the hardware and infrastructure, such as storage and network devices, sensors, transducers, and wired or wireless communication links between them. Physical components are prone to physical damage since the physical layer itself has a grounded sense of location. As a result, physical security measures and mechanisms need to be employed to prevent unauthorized access to this layer [45].

2. Logical Network Layer

The logical layer consists of the software and its manifestations, and the network elements that are code-related and make use of the physical foundations [45]. The decisions made at this level define the nature of cyberspace, its strengths, and its limitations. How the network is built, its capabilities, and why vulnerabilities exist constitute part of the logical layer. Within this layer, we observe sub-layers that provide services to the layers above them, while they are designed to be combined to offer more complex services. For example, some low-level services are the mechanisms for data transport. Out of these are built applications like a database or web pages, and if we combine these, we get dynamic content generation and active web objects.

At this level, cyberspace is a series of stacked platforms, one being the foundation for the next one above it. New capabilities are established on each platform in response to the continuous evolution of cyberspace. The nature of cyberspace is thus characterized by the rapid evolution of new services derived and combined from logical constructs, all operating on top of physical grounds [46].

3. Cyber-Persona Network Layer

The cyber-persona layer consists of the active participants of cyberspace, i.e., the system users, whether human or automated, and the connections between them. A user's digital representation in cyberspace (their cyber-persona) can relate to another entity or human making use of personal data, like IP address and phone number. A single user can have multiple cyber-personas using different credentials or accounts, and likewise, one cyber-persona might have multiple users by using, for instance, a group email address [45]. See Figure 4.

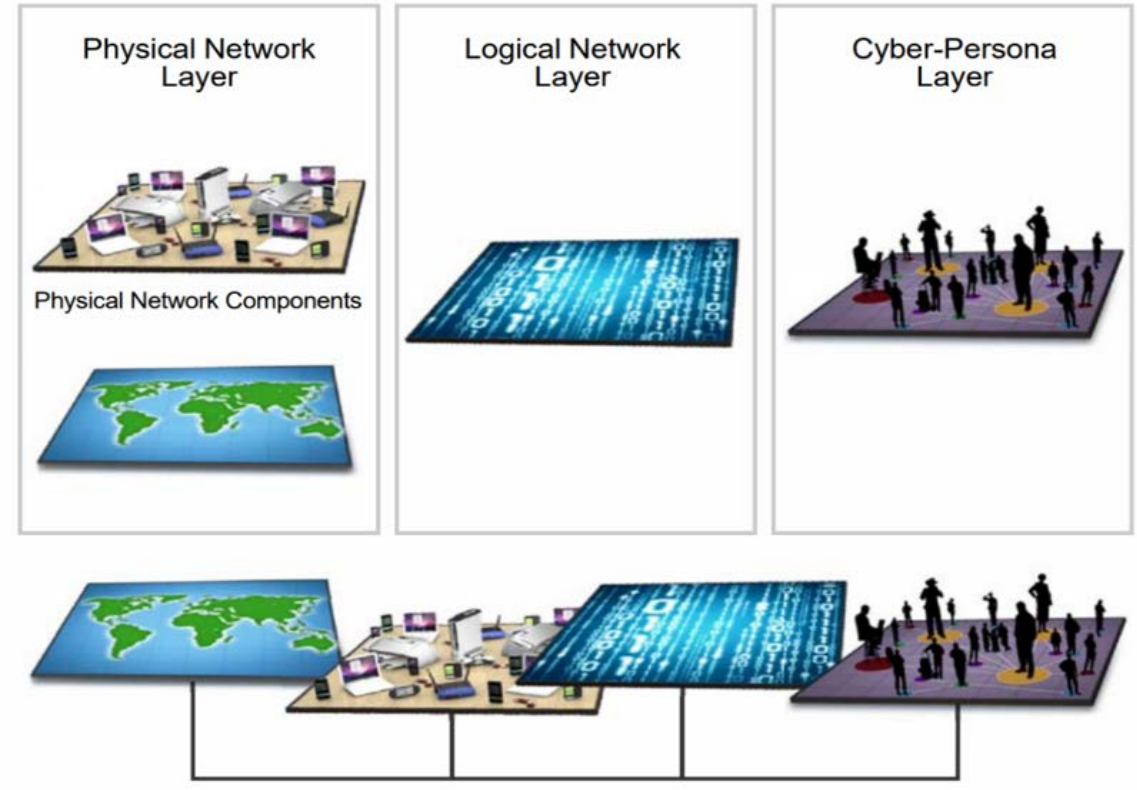

Figure 4. The Three Interrelated Layers of Cyberspace. Source: [45].

Although JP 3-12 describes three network layers, other reports describe more. This could be the result of misunderstanding the concept or probably having a different perspective and giving a different definition. Choucri et al. added the information layer between the logical and physical layers to refer to the processing, storage, and distribution

of information in every possible form within cyberspace, pointing out that one with a different vision could build a totally different system [46]. Raymond et al. [47] used five cyber planes as a framework to describe cyberspace, with real-world examples in each plane. Three of their five cyber planes/layers correspond to the three network layers in JP 3-12. The other two layers are the supervisory plane that provides the authority to manage a cyber operation, and the geographic plane that describes the area where an IS is located.

C. WELL-SECURED INFORMATION SYSTEM CONFIGURATION

The process of IS configuration has a significant role in InfoSec. The poorly configured IS is one of the most common ways that a malicious adversary may take advantage of launching a cyber-attack against the system. A well-secured configuration defines all the security measures that need to be implemented when building and installing an IS. The purpose is to reduce the existence of rectifiable cyber vulnerabilities in the system. Unfortunately, as an organization grows, their technology landscape becomes increasingly more sophisticated and complex, and they need to protect against failures across multiple areas. The components that need to be configured could be an individual object or a collection of them where the complexity is escalated. These elements are often hierarchically structured, so the overall security should be achieved at each level. For this reason, we need a process of documenting system changes that:

- Monitors and records the functional and physical characteristics of an object and system as it evolves over time;
- Presents, at any time and in detail, the evolutionary history of the system as well as the present situation, and;
- Ensures that all components of a system follow the rules.

If these criteria are not implemented, we deem an IS to be poorly-configured. The first step for configuring a well-secured system is to set a gold standard configuration, and continual monitoring of the IS against this standard [48], [49]. There are several standard frameworks in the industry, such as NIST, CIS, ISO, etc. These are a set of criteria known as the best practice that an organization should follow. The implementation of a standard

framework is a demanding procedure that includes the execution of specific steps to achieve the development of a secure integrated configuration. Those steps can be met in any industry-standard framework, and we will summarize them to the following five steps:

1. Step 1: Current Situation Imprinting

In the first phase, we need to record all IT assets of the target organization. These assets could be divided into several categories, such as data assets, end-user services, materials data, locations, software, etc. After capturing the organization's asset status, the role of the users should be defined, and each user should be classified. This step is crucial because the assets should not be examined only as individuals but also as a part of a set. The interaction of the assets with each other has an impact on the complexity of the system. It is hard to maintain security and availability, and with each new user, device or application, the volume of what needs to be monitored and protected increases [50].

2. Step 2: Risk Analysis

Following the precise evaluation of the target's assets, a risk analysis is performed. This is a particularly important step in shaping the target's security policy. In this phase, there is a detailed study of exposures of the system, and a determination is made of the vulnerabilities and threats of the system based on the current controls. The method used to conduct the risk analysis may be based on the formula BPL ($B > P * L$), where B is the cost to prevent a loss, P is the probability of a loss occurring, and L is the total cost of a loss [51]. The cost of preventing a loss or damage will, therefore, be compared with the cost of such a loss, together with the likelihood of such a loss occurring.

Then, the threats for each asset and their impacts are recognized. Additionally, potential vulnerabilities for each asset separately identified. Possible losses are recognized and categorized according to the degree of risk. The likelihood of a loss occurring is also assessed while identifying the necessary countermeasures to avoid risks. At this stage, those countermeasures that are most advantageous to the organization are selected. Finally, the most appropriate and efficient countermeasures are configured and implemented [52]. The effectiveness of countermeasures should be continuously monitored because of the constant change of data (new threats, new vulnerabilities, new impacts, etc.) in an IS. The

completeness of the risk analysis of an IS is associated with the safety of the physical, logical, and cyber-persona layers.

3. Step 3: Security Plan

An IS security plan (SP) provides guidance and support on information security issues. The SP is determined by the organization management and should be supported in practice by the organization itself. The SP should regulate security issues at all levels of the organization, and record and elaborate on the set of laws and rules that govern how data is managed, secured, and allocated within an organization. The formulation of the SP should adopt international standards, in particular the Trusted Computer System Evaluation Criteria (TCSEC) and Information Technology Security Evaluation Criteria (ITSEC). TCSEC and ITSEC are standards of U.S. DOD and Europe, respectively [53]. They are a structured set of criteria that are used to rate the efficacy of computer security within products and systems. The organization's management should accept and approve the text of the security policy. Then, it should be made public to all users of the organization, so that everyone will know about the administration's commitment and how the organization approaches security issues. This step is an additional contribution to the risk analysis because the SP is one of the most critical countermeasures against the threats that were identified in the previous step.

4. Step 4: Determination of Security Measures

In this step, the SP is implemented by designing measures that will meet the system's security requirements. Management and planning the security of the IS are related to the code of conduct of the organization, the security controls, and the roles and responsibilities of the security administration. Equally important is to educate and inform the users about the procedures and, more generally, about the security functions of IS [50]. Finally, we should consider the security of the backup process and the confidentiality of the violation records. This step should include periodic reviews and revisions of the measures to avoid an overlap between them.

5. Step 5: Contingency Plan

The IS contingency plan (CP) works in conjunction with the SP. It records all procedures and implements measures that ensure the uninterrupted functionality of the organization in a possible disaster, malfunction, or total destruction. Overall, the CP not only includes the recovery of the computing function and communication infrastructure, but it also includes information and procedures for a spare installation (disaster recovery facility), or an available alternative location (alternate site) for staff relocation in case of total infrastructure destruction, where it is necessary [53].

A well-secured configuration is the first step for hardening an IS to reduce the attack surface and mitigate the damage of an attack. In the next section, we discuss the importance of a well-secured system by examining the ransomware attack of 2017 known as WannaCry [54]. The effects of combining a zero-day vulnerability and a poorly-secured or out-of-date configured IS could be like the effects of the tsunami; they are catastrophic and rapidly spread.

D. ZERO-DAY VULNERABILITY CASE STUDY: WANNACRY RANSOMWARE ATTACK OF 2017

In May 2017, a worldwide cyberattack was launched using the WannaCry ransomware attack, which targeted computers that were running a Windows operating system with Server Message Block version 1 (SMBv1) [54]. This ransomware attack encrypted files on the victim's computer, then demanded money as Bitcoin in ransom for exchange of a decryption key, which allowed the user to access the encrypted files. The attack was also considered a web virus because it contained a transport mechanism that allowed it to replicate itself to other victims. In essence, the transport code scanned the network for vulnerable computers, then used the EternalBlue [55] exploit to access them by sending crafted packets from attackers, allowing them to execute arbitrary code remotely. And finally, the DoublePulsar [56] tool, which runs in kernel mode, and was used as a backdoor through which the virus was injected and installed [54].

WannaCry exploited a vulnerability in the Windows SMBv1 protocol to compromise computers, loaded malware, and transmitted it to other computers on a target

network. WannaCry utilized the vulnerability of allowing remote code execution by crafting a custom session request, which included a hard-coded domain name. Before any operation, a query to that hard-code domain name was initialized; a successful connection caused the termination of “mssecvc,” otherwise it was proceeded [57]. This attack used the SMB version 1 and TCP 445 port to propagate in the network. The use of SMB transactions allowed for individual reading and writing between an SMB client and a server.

After the initial SMB handshake, the ransomware connected to the shared inter-process communication (IPC) $ on the remote machine. One of the features of the mentioned cyberattack was that the malicious software was programmed to establish a connection to a “hardcoded” network identity. It then sent an initial “NT Trans” request. In the case that the request was larger from what is allowed by the SMB “MaxBufferSize,” the rest of the request would be transferred as subrequests, resulting in a buffer overflow. This vulnerability affected the “srv2.sys kernel” [54] and was set off by distorted “Trans 2 secondary requests” [55]. These requests included a large payload, consisting of a number of NOPs (No-operation instructions), as presented in Figure 5. The malware was moving the current status of the SMB server to a state where the attacker could take advantage of the existing vulnerability and attack it with a specially designed package [54].

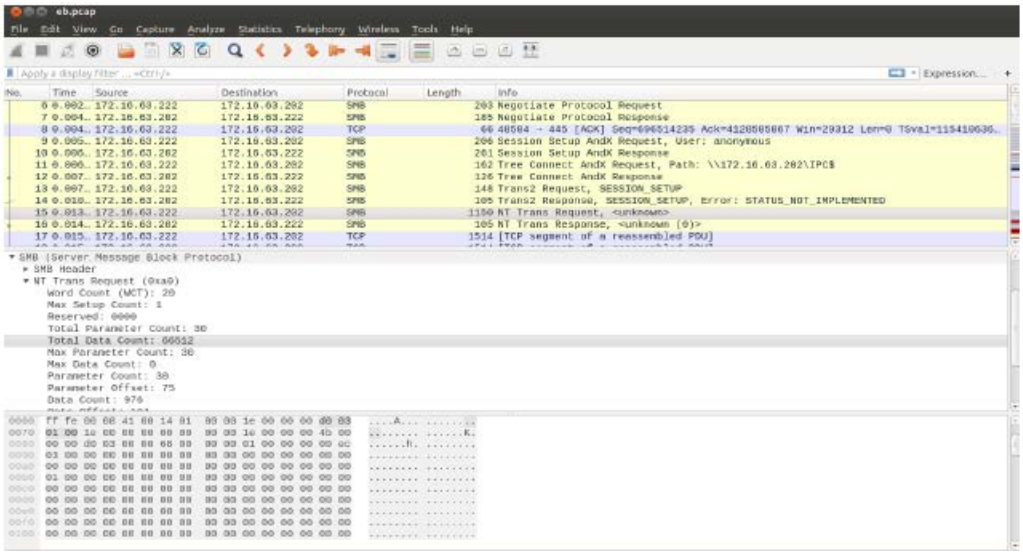

Figure 5. Preparing Attack via NT Trans. Source: [58].

After successfully exploiting the vulnerability of the protocol, an encrypted "payload" has the stager of malware, is transferred in the victim's computer remotely. This "payload" launched a service on the computer called "mssecvc" [59] through the lsass process. Lsass is a process of the Windows operating system that enforces system security policy. It also verifies users logged on to a Windows server or computer, manages password changes, generates access tokens, and writes to the Windows security log. The "mssecvc" service was acting by scanning not only the local network but also the internet for devices which were accessible, and their SMB ports were exposed. Then it used the vulnerability as mentioned earlier to gain remote access and infected them with the malware.

The effects of this attack were massive, and they were rapidly spread. According to Europol data, more than 200,000 computers were infected in 150 countries. The National Health Service hospitals in the UK were the institutes that were affected by the malware mentioned above. It was found that over 70,000 devices were infected, including medical equipment such as MRI scanners, etc. The financial damage caused by WannaCry is estimated to be more than $4 billion [54].

Microsoft discovered the zero-day vulnerability exploited by WannaCry on March 14, 2017, and they released a security patch nearly two months before the attack was launched. As seen in the attack timeline in Figure 6, the extent of the infection could have been prevented if IS administrators had installed the patch that Microsoft released on March 2017, and if there were not systems that were still running out of date products of Microsoft [58]. Additionally, we should point out that the application of the released patch would have helped to the mitigation of the spread between poorly configured systems but would not have prevented the infection.

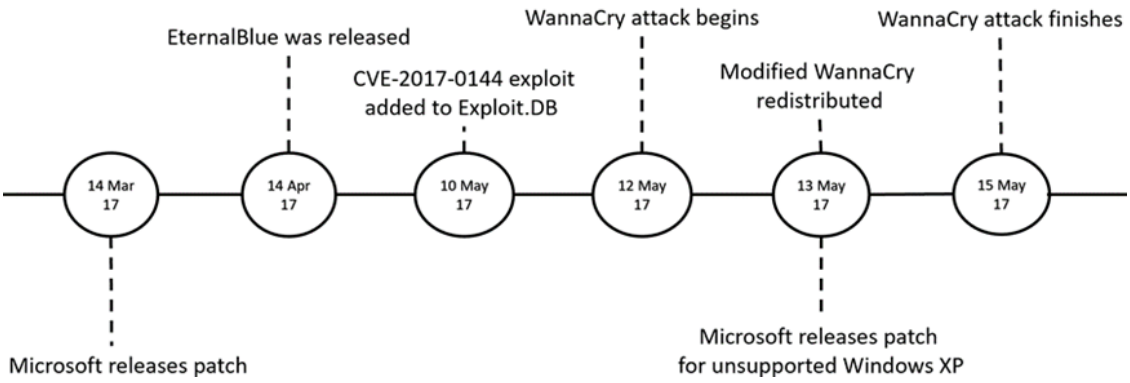

Figure 6. Timeline of WannaCry Attack

E. CHAPTER SUMMARY

In this chapter, we defined the challenging issue of information system security, and we presented the steps for a well-secured IS configuration as they are defined in several industry-standard frameworks. Next, we examined the WannaCry attack of 2017, which pointed out the potency of a zero-day attack and the escalation of the results on a poorly configured system. Moreover, it demonstrated that in a modern world where new zero-days vulnerabilities are revealed every day, a well-secured configuration is indispensable.

The next chapter covers a zero-day attack from the defender's and the offender's perspective. It examines the impact of a zero-day vulnerability in the cyber defense of a system. Additionally, it analyzes the life cycle of exploit development and the factors that can affect it.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ZERO-DAY ATTACK FROM THE DEFENSIVE AND OFFENSIVE PERSPECTIVES

The exploitation of a zero-day vulnerability is considered the "holy grail" from the offensive cyber perspective. An adversary can launch a zero-day attack, i.e., an attack specifically targeting a zero-day vulnerability, leaving the defender in a disadvantageous position and called upon to defend their system against an attack that was previously unknown.

This chapter focuses on how the window of exposure for a zero-day vulnerability impacts the cybersecurity of a system. Also, we suggest countermeasures and guidelines to reduce the possibility of the impact of a zero-day attack. We examine the exploitation of zero-day vulnerabilities from the offensive perspective, including the life cycle of exploit development and their use in the wild, as well as factors beneficial to exploit development.

A. DEFENDER PERSPECTIVE

Robert Mueller, in 2012, said, "There are only two types of companies: those that *have been* hacked and those that *will be*" [60]. With this in mind, an organization should plan their cyber defense actions as if the breach has already happened. As discussed in the previous chapter with the WannaCry attack of 2017, the effects of a zero-day attack can easily become out of control and their impact catastrophic.

1. Impact on Cyber Defense

In the timeline of a zero-day attack, the most crucial period for an organization is the time between the introduction of the zero-day vulnerability and the time that an effective patch is completely deployed. This period is known as the "window of exposure," and it determines the security risk exposure of the organization's overall cybersecurity [61]. During this period, an organization is vulnerable and the only mechanisms it has for its defense are the security plan (SP) and security measures, and the contingency plan (CP), described in the previous chapter.

The impact of a zero-day attack depends not only on the kind of zero-day vulnerability that was exploited, but also on the structure of the system that was targeted. Systems that have implemented defense-in-depth could restrict the effects of a zero-day attack because of their multiple defensive layers that are built-in to protect the cyber infrastructure [62]. If a zero-day exploit bypasses one defense mechanism, there are several others in place that could potentially thwart the zero-day attack. Stated plainly, strong defense-in-depth may be able to indirectly withstand such attacks, despite a defender's lack of knowledge about the zero-day vulnerability being exploited in the attack.

The existence of a zero-day vulnerability could expose the entire cybersecurity of a system. An adversary could be targeting confidential data, financial information, or critical business plans, which could cause an intolerable effect on the operation or the reputation of the organization. An indication of an organization's ability to minimize the impact of a zero-day attack is closely tied to two criteria: how quickly the vulnerability is handled, and how quickly a patch is developed. This reaction time is also known as zero-day patch rate, which is "the number of patches a vendor is able to release at the day of the public disclosure of a new vulnerability" [61]. By having a high zero-day patch rate, vendors try to reduce the window of exposure for their users. However, in order to achieve this, the vendor will need advance notification and a vulnerability research team that will continuously investigate their product or by cooperating with independent vulnerability researchers [61].

Several studies, such as Shahzad et al. showed that OS and products from Microsoft and Apple are more attractive to hackers [1], [3], [63]. Therefore, it is more rewarding for attackers to develop a zero-day exploit for products from these two companies because of their popularity in the market and the high use in enterprises. Additionally, the comparison between closed-source and open-source vendor patch rates showed that open-source vendors have a slower response as they rely on volunteer developers to investigate their code and disclose zero-day vulnerabilities [1]. At the same time, an adversary could more quickly develop a zero-day exploit against an open-source product due to easier access to the code. It should also be noted that different vendors show better performance in different kinds of zero-day vulnerabilities; for example, the best choice for an organization that

needs to harden their system against buffer overflow attacks should be the deployment of the Solaris OS, which has a statistical buffer overflow vulnerability rate of only 13%, as opposed to a 20% rate in Windows and MacOS systems [1]. To conclude, although the statistics point out that most zero-day exploits were targeting closed-source products, an organization should not rely only on that for its cyber defenses. Robustness cybersecurity should take in account the characteristics of a network and the zero-day patch rate that each vendor offers.

2. Characteristics That Increase the Impact of a Zero-Day Vulnerability

There are characteristics of a network that can increase its vulnerability to zero-day attacks. The complexity of the network is a common factor that affects the management efficiency of a system. The topology, in combination with the complexity, could make an entire network more vulnerable to zero-day attacks, because of the variety of different components that need to be integrated, which could create inconsistencies in overall security. So even if it is widely believed that "greater diversity in software and services may help to improve a network's security," the implementation should be undertaken with precautionary steps [64]. The heterogeneity of network component functionalities and operating conditions could create security gaps and potential new zero-day vulnerabilities that did not exist in each individual component.

Networks consisting of IoT devices are more vulnerable to zero-day attacks, especially when these devices are not divided into separate zones. The implementation of firewalls and Demilitarized Zones (DMZs) can lead to better security [65]. IoT devices should be segmented from critical infrastructure because they have limitations in the hardware and lack of computational power. So, often they do not have built-in security, and for that reason, they can more easily be hijacked. Moreover, many IoT devices use firmware that could be found on the Internet, and an adversary can easily examine it for potential flaws that can lead to exploitable zero-day vulnerabilities.

The strategy to segment the network should also be followed by organizations that allow their users to "bring your own device" (BYOD). If users misuse their devices or lose them, an adversary could take advantage of them to create a breach in the security of the

organization. From a security administrator perspective, BYOD could increase the potential access points for zero-day attacks especially if users do not follow security guidelines and requirements. Therefore, separating the network into zones could improve network's security. In this way, the effect of a breach could be minimized to only a specific segment, and any critical zones could be unaffected by placing them further from the perimeter. A suggested strategy for the administrator is to use DMZs to restrict access from the internet to the internal network of an organization. In this way, even if an adversary has developed a zero-day exploit for the targeted system, it still must penetrate through the first layer of security, the DMZ. Next, if the adversary is able to penetrate the DMZ, a well-configured deny-by-default firewall that whitelists only known safe traffic will be the second obstacle to overcome. So, by dividing the network to different zones and filtering the traffic between them, an administrator empowers the organization's security.

3. Zero-day Countermeasures

Although defenders have many solutions to guard their systems against known attacks (e.g., IPS/IDS, antivirus, firewall, etc.), zero-day attacks present unique challenges. The defenders are in a disadvantageous position since the lack of knowledge and information about these unknown attacks deprive them of the opportunity to defend their system effectively. For this reason, a defender's goal should be to reduce the attack surface of the system and built their defense-in-depth with several layers. At this point, the prevention and not the detection of a threat is the most efficient solution to mitigate the results of a zero-day attack. The following sub-sections describe countermeasures and guidelines that can increase a defender's zero-day vulnerability awareness.

a. Address Space Layout Randomization

Address Space Layout Randomization (ASLR) is an operating system framework that improves security by randomizing address space allocation in memory. ASLR can be used by the stack, the heap, the program's pile, or a DLL, and each time a memory address is assigned, it is different from that which was used the previous time [66]. By randomizing the memory location of a process in memory, ASLR prevents an adversary from being able to assume the memory address of the targeted vulnerability. ASLR is a security tool that

can prevent the exploitation of a zero-day vulnerability by obfuscating the mechanism for allocating memory and adding a layer in the system's defense [67]. So, ASLR makes zero-day vulnerability exploitation more difficult for the adversary, without having to remove the vulnerability.

b. Host Intrusion Prevention System

The purpose of Host Intrusion Prevention System (HIPS) is to secure a system against known malicious activities by detecting their signatures. Some HIPS systems are anomaly-based, meaning they use an acceptable range of "normal" activities and behaviors to raise alerts when this range is violated. HIPS can monitor the host and search for anomalies in the behavior of system activity. HIPS operation is not determined by any specific signature, it uses a baseline for the activity of the ports, processes, protocols even in the bandwidth [68]. So, in the case of malicious activity, a HIPS can stall this activity and alert the administrator to its occurrence. If malicious activity causes abnormal behavior observations, HIPS could detect it so that it could be successfully used as a tool to identify and protect against certain zero-day attacks. Moreover, it provides an extra customizable layer of defense because it allows the administrator to parameterize the setting for each system [67]. The "normal" activity is not the same for each host in a network, so by customizing the rules for each host, the benefit for an administrator is that the rate of false-positive alerts is decreased.

c. Stateful Application Control

Stateful application control continuously examines and validates the current application state against the database of all proper known application states. Stateful application control does not need specific malware signatures or information about malicious activity, therefore, it is an efficient way to block zero-day attacks. Its operation depends on validating the state of the application, and in this way, it can successfully block unknown threats. Stateful application control monitors the state whenever a connection is established, or an operation writes to the file system [69]. An application is allowed to proceed if and only if its current state matches a legitimate state, which means that the data of the executable procedure are validated. However, if the state does not match, it is an

indicator that malicious activity may be taking place, and the machine should be protected from potential risk [69]. Stateful application control will block the execution and will alert the administrator regarding the trace of a suspicious attempt. Therefore, it provides the ability for an organization to protect itself against known or unknown threats.

d. Secure OSI Layer 8

Even if most of the cyberattacks are highly sophisticated and use advanced techniques, it should never be forgotten that the untrained human is the weakest point in the chain of security. Sometimes it is easier to exploit a system by taking advantage of human errors. Therefore, the education of the user should be a high priority in an organization. Users should establish security habits and best practices that will keep them safe, and consequentially an organization will build an extra layer against zero-day threats [70]. The education of the user should make them aware about social engineering techniques that could target them and how they should act if they have a flicker that someone is targeting them or trying to compromise them.

B. OFFENSIVE PERSPECTIVE

The success of a cyber-attack is more likely if the vulnerability it exploits is not known to the targeted victim. According to the recent research, once the vulnerability researchers discover an exploitable vulnerability, the average time to develop a functioning exploit is relatively short, with a median time of 22 days, and it can survive on average between 5.39 and 8.84 years [3]. This seems to indicate that attackers have an advantage over defenders because they can develop a zero-day exploit in much less time than it takes for defenders to become aware of the zero-day vulnerability and develop a patch to mitigate it.

1. The Life Cycle of an Exploit Development

Research by Ablon et al. showed that the procedure for the development of an exploit includes many steps, each of which can go through several iterations, as shown in Figure 7 [3]. The exploit development life cycle is composed of three phases “Discovery,” “Implementation,” and “Operational Handoff”.

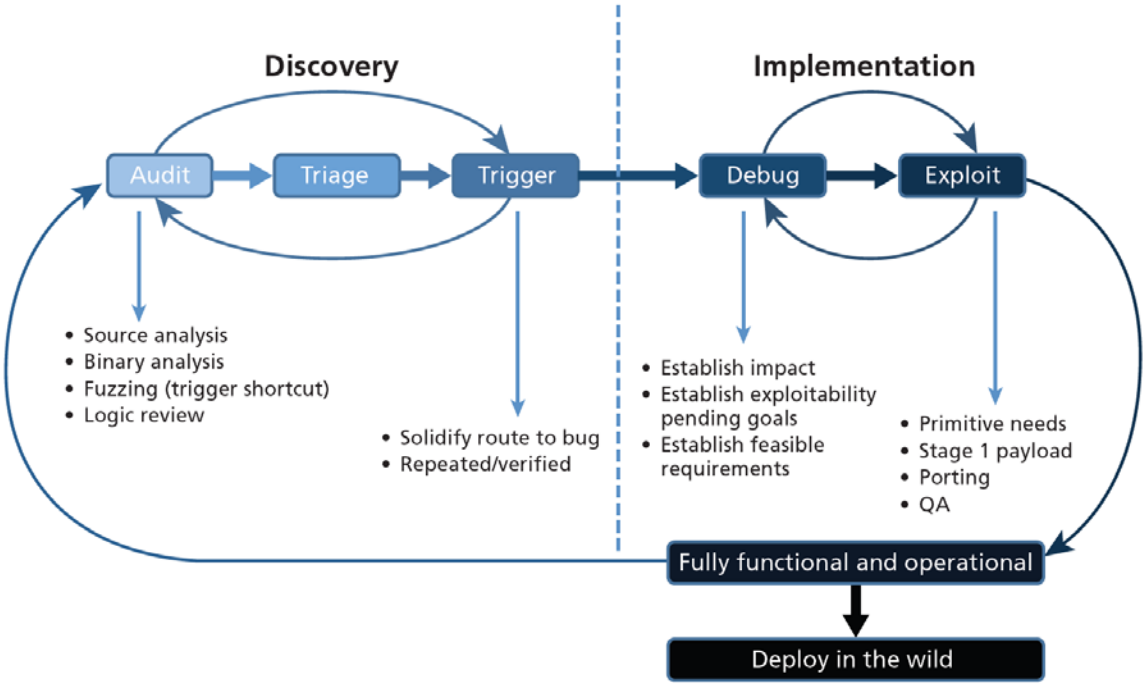

Figure 7. The Exploit Development Life Cycle. Source: [3].

The "Discovery" phase aims to discover any unknown vulnerabilities and built a Proof of Concept (PoC) exploit. This phase includes the following steps [3]:

- **Audit.** The vulnerability researchers try to crash the vulnerable system by triggering errors under abnormal operating conditions. The complexity of the code and compatibility issues usually offer the researchers more opportunities for identifying exploitable vulnerabilities.

- **Triage.** The crash is examined to determine the root cause. The cause of the crash is valuable because it could reveal a potential zero-day vulnerability.

- **Trigger.** Once the vulnerability researchers have determined the cause of the crash, they will need to develop a reliable procedure to trigger it and provoke a crash at will. This stage is where a PoC exploit is created.

The **Trigger** step is the final one for researchers who work for the benefit of the vendor, such as bug bounty hunters. They stop at the end of the "Discovery" phase because their intent is to identify and understand the conditions that actually trigger the exploit. For

zero-day exploit developers, however, they transition into the “Implementation” phase, where their goal is to have created a fully functional exploit for the target system. Once again, this phase is iterative, and its steps are [3]:

- **Debug.** The researchers examine each crash, evaluate the reliability of the trigger causes, and determine how they should take advantage of those causes. If more requirements are needed for the development of a fully functional exploit, the previous phase should be repeated.

- **Exploit.** Here the researchers determine the components for the creation of a fully functional exploit. They will focus on the establishment of a foothold, and build up the exploit by adding in capabilities to increase its reliability.

The final phase of the exploit development life cycle is the “Operational Handoff” in which the vulnerability researchers transfer their developed exploit from lab to a real-world scenario where the environmental conditions may vary each time. After a successful launch of the exploit, the developer needs to change the signature of the exploit for follow-on attacks to avoid detection by AV and IDS security systems. This will increase the window of exposure and the longevity of a zero-day vulnerability [3].

The most intriguing correlation of previous analysis about the development of a zero-day exploit is the result of the statistics about the development time of exploits and their longevity. It demonstrates that the offensive side had the upper hand when they discover a zero-day vulnerability before the vendor or system defender. Even though the procedure of the development of a zero-day exploit seems to be time-consuming, the statistics confirm the opposite when the range of the development for the majority of exploits is between 6 and 37 days [3]. This highlights the advantage that an offender has against the targeted system, since they have time to research, test, and reinforce an exploit before the vulnerability it targets becomes known. At the same time, the lack of a patch or any detection mechanism for the vulnerability will not prevent development of a reliable and efficient zero-day attack.

Additionally, successful completion of the “Discovery” phase could not only reveal new zero-day vulnerabilities but provide ways to explore zero-day vulnerabilities that were

unexploitable until then. Also, it may be assumed that in a system where one zero-day vulnerability is discovered, multiple vulnerabilities could be found.

Another striking observation is the comparison of the average life of a zero-day exploit (6.9 years) [3] to the average time that an identified zero-day vulnerability remains publicly unknown (310 days) [29]. This seems to clearly demonstrate that exploiting a zero-day vulnerability is far easier (or at least takes far less time) than publicly disclosing and patching it by the vendor. Therefore, the adversary could patiently follow the procedures of exploit development, take advantage of the fact that the vendor is unaware of or ignores the existence of the zero-day vulnerability, and take the needed time to develop a fully capable and effective exploit. Moreover, by examining the average life of a zero-day exploit, it can be reasonably assumed that developing a patch is demanding and needs a sophisticated approach. Generating a patch requires reverse-engineering of the zero-day attack and patching the root cause for every way that it could be possibly be triggered.

Finally, the critical element to the success of a zero-day exploit is the length of its development period, which overlaps with the period between vulnerability introduction (t_v) and exploit released in the wild (t_e), as illustrated in Figure 2 in Chapter 2. Several factors could impact the exploit development period, such as the skills of the developers, their resources, defender security mitigations, and vendor awareness, etc.

2. Factors That Impact Exploit Development

Unfortunately, there is a lack of research in real data of zero-day vulnerabilities. As Ben Hawkes presented during his presentation at the Black Hat USA in 2019, “the value for the attacker of a zero-day outweighs the value for the defender. The attacker is incentivized to keep information private and mislead us” [71]. This is the reason for the existence of a black market for zero-day vulnerabilities, which can be sold for $50,000-$300,000, depending on the kind of the zero-day vulnerability [3]. Some ways that may be beneficial to the OCO operators for reducing the total time to develop a zero-day exploit are suggested in the following sections.

a. Training

Several researchers at the Black Hat conference in 2018 discussed their first steps in the research of zero-day vulnerabilities and how they developed their skills. One common characteristic was their participation in Capture the Flag (CTF) and war games. They mentioned that this helped them to sharpen their skills and helped them identify patterns in the discovery of vulnerabilities since, as they noted, vulnerabilities in the same class have the same pattern [72]. Additionally, another approach could be the use of machine learning, where artificial intelligence (AI) can be trained from CTF and war games data. In the future, the AI approaches may become powerful assets in the research and discovery of potential zero-day vulnerabilities. There is a parallel one can draw between this and the approach followed in Mayhem [73]. Mayhem did not specifically focus on finding zero-day vulnerabilities but his approach could be extended to this class of vulnerabilities.

b. Stockpile

After the discovery of a zero-day vulnerability, researchers need to determine if they can develop an exploit. Only a subset of zero-day vulnerabilities can lead to a fully functional exploit. Many times, the exploitation of the vulnerability may need specific circumstances, or a "blocker bug" may protect the target system and stand in the way of developing a zero-day exploit. In this case, researchers may keep such vulnerabilities hidden for future use, because something that is not exploitable one day may be in the future. This is known as *stockpiling* [3]. The RAND Corporation, using real world data from zero-day vulnerabilities, showed a low collision rate to undiscovered vulnerabilities [3]. This showed that the likelihood of another vulnerability research team discovering the same vulnerability, and presenting a risk of losing a zero-day vulnerability from the stockpile, is slight. So, the researchers should revisit them frequently when a new version of the target system is released.

Another benefit of zero-day vulnerability stockpiling could be the development of an exploit that takes advantage of more than one vulnerability on the target, even if they are known vulnerabilities [74]. Research could reveal combinations that will increase the reliability of a zero-day attack and reduce the development time because of the existing knowledge.

c. Weakest link in the chain

The desire to automate systems and reduce human intervention has resulted in the broader use of automated systems on many organization's network. Such systems can be part of the Internet of Things (IoT) that transfers data over the network without the interference of a human. The security of such systems can be described as a chain, which is only as strong as its weakest component.

Many IoT devices use universal firmware, which makes it easier to manage and control multiple IoT devices. This use of common firmware subjects the IoT devices to several types of threats [75]. An adversary can take advantage of these threats to save time and bypass the device's security or test them for potential zero-day vulnerabilities. Additionally, an adversary can exploit characteristics of the device that are more frequently used in working zero-day exploits, such as memory corruption and deserialization [3], [71]. The strategy of identifying the weakest link in these systems, and testing it for the most common characteristics of zero-day vulnerabilities, could minimize the development time of a zero-day exploit.

C. CHAPTER SUMMARY

In this chapter, we examined zero-day attack from the defender and the attacker perspectives. We evaluated the impact of zero-day vulnerabilities on system cybersecurity and analyzed potential characteristics that make a system more susceptible to zero-day vulnerabilities. Moreover, we presented standard countermeasures and guidelines to increase the zero-day vulnerability awareness of the defender. Finally, from the attacker perspective, we analyzed the life cycle of exploit development and suggested measures that would be beneficial in the development time of a zero-day exploit. The next chapter provides overall conclusions of this research and discusses potential opportunities for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. SUMMARY

The main goals of this thesis were to thoroughly examine the importance of zero-day vulnerabilities to cybersecurity and defense, and to analyze how the development time of zero-days exploits can impact offensive cyber operations. We conducted background research to more thoroughly understand the crucial role of zero-day vulnerabilities in cyber operations, both offensive and defensive, and we analyzed the life cycle of zero-day vulnerabilities to understand how it affects the cyber defense of a system.

We then examined the dynamic concept of information system security and the implementations of several industry-standard system frameworks to suggest well-secured IS configurations. This concept is critically important because a well-secured IS configuration is the first obstacle that an adversary will confront in trying to penetrate the system, so it can greatly limit the attacker's options and mitigate the results of their actions. As a demonstration of this, we studied the results of the WannaCry attack of 2017.

Additionally, we examined zero-day attacks from the defender's and the offender's perspectives, specifically the impact of a zero-day vulnerability to cyber defense of a system. We presented potential characteristics that make a system more vulnerable to zero-day vulnerabilities, and countermeasures and guidelines to reduce the attack surface of such a system to mitigate the damage of a zero-day attack against a zero-day vulnerability. Finally, we studied the life cycle of exploit development, and how awareness of zero-day vulnerabilities can be beneficial to the development time of a zero-day exploit. These ideas are aimed to reduce the development time of a zero-day exploit and benefit OCO operations.

B. CONCLUSIONS

The final conclusions to this thesis are derived by answering the original research questions posed in Chapter I.

1. Primary Question

How does the existence of zero-day vulnerabilities in computer systems, and the time for adversaries to develop exploits against these vulnerabilities, impact the overall level of cybersecurity of that system?

We reasoned that a zero-day vulnerability in a system can compromise the entire security infrastructure of that system. Its overall impact is not only determined by the kind of zero-day vulnerability, but also by the structure and configuration of the targeted system. Systems with layered defenses may be able to better restrict and mitigate the effects of a zero-day attack because multiple layers of defense force vulnerability exploiters to penetrate through each layer.

Additionally, we found that systems with higher complexity are more likely to be found with zero-day vulnerabilities or ways to exploit zero-day vulnerabilities that were previously not exploitable. Moreover, systems that do not implement segmentation of the network to separate IoT devices and user systems from the critical infrastructure offer a larger surface for zero-day vulnerability exploitation. Finally, systems with a high zero-day patch rate tend to reduce the time of the exposure window and, therefore, minimize the overall impact of such vulnerabilities to their cybersecurity.

2. Secondary Question

How might the total time to develop zero-day exploits, including the time to identify associated zero-day vulnerabilities, be reduced to better support offensive cyber operators?

In studying statistics of the life cycle of exploit development, namely the average time to develop a zero-day exploit, the average time that a zero-day exploit survives, and the average time that a zero-day vulnerability remains publicly unknown, we conclude that a zero-day exploit can be developed more rapidly than the time it takes to discover and patch a vulnerability. This is because the average time for an adversary to develop a reliable zero-day exploit is 22 days [3], while the time for a vendor to identify a zero-day vulnerability that it is already discovered by an adversary is about 310 days [29].

A complete answer to this research question is somewhat limited by the lack of research on real-world data on zero-day vulnerabilities. The high value that zero-day vulnerabilities have for attackers and the cyber black-market motivates researchers to keep their results private, which makes open academic research difficult. However, from our research we can conjecture several factors that could potentially reduce the development time of a zero-day exploit and benefit OCO operators:

- Training zero-day vulnerability research teams in Capture-the-Flag (CTF) and similar cyber war games can help sharpen their skills, expand their CO toolbox, and identify patterns in the discovery of same class zero-day vulnerabilities. A quite promising approach would be to analyze data on how participants in CTF and war games are working to discover vulnerabilities and use them for training an artificial intelligence (AI) system.

- By stockpiling zero-day vulnerabilities that could not be exploited at the time they were discovered, they can be kept hidden for future use since they might be useful in a future scenario or may become exploitable under different circumstances.

- Targeting the weakest link in the security chain. The increasing usage of IoT devices and sometimes their lack of built-in security or proper configuration makes them a valuable target for exploitation. In particular, focusing OCO efforts on discovering zero-day vulnerabilities in these devices offers a potentially rich source of exploitation.

C. FUTURE WORK

The following presents areas for future research to expand on the work done in this thesis.

1. Automated Zero-Day Vulnerability Analysis Tool

This research was a first step towards deepening an understanding of zero-day vulnerabilities and their exploitability. A goal of the research was to provide a foundational work toward creating an automated zero-day vulnerability analysis tool. This automated tool would scan for zero-day behaviors in systems to identify potentially unknown vulnerabilities. The use of this tool could be beneficial either for the vendors to help them identify zero-day vulnerabilities in their products and patch them, or to OCO operators to help them discover exploitable zero-day vulnerabilities on targeted system.

2. Examine Open-Source Products and Compare Them to Closed-Source

Many organizations today widely use open-source products for their business practices and procedures. Future research could focus specifically on examining whether open-source products are more beneficial for system defenders or their attackers. It should question if the openness of a system helps the vendors discover quicker a zero-day vulnerability, and publicly disclose it and patch it before an adversary can exploit it. Also, it could examine whether the open-source nature of such systems enhances their security by allowing more people to contribute to their defense.

3. AI And Zero-Day Vulnerabilities

As it was mentioned before, the usage of AI would be a powerful asset in the research and discovery of zero-day vulnerabilities. Data from CTF and war games could be collected and used for the training of AI. Later, the AI could be tested in a system for discovering zero-day vulnerabilities and then compare the result to the result of a human vulnerability research team.

4. Study Real-World Data of Zero-Day Exploits

Further analysis needs to be done on real-world data of zero-day vulnerability exploits to establish better awareness about them. Knowledge of zero-day vulnerabilities is a great asset for attackers as they can profit by selling this knowledge in the cyber black market. This can dissuade researchers from publishing their results, which makes open

academic research difficult. The output from an analysis of real-world data of zero-day vulnerability exploits would be beneficial for the defender's perspective. Defenders and vendors could reduce the impact of a zero-day vulnerability and improve their methodology to discover and patch it. This requires detailed knowledge of how attackers work to reinforce their defenses.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] M. Shahzad, M. Z. Shafiq, and A. X. Liu, "A large scale exploratory analysis of software vulnerability life cycles," in *2012 34th International Conference on Software Engineering (ICSE)*, Jun. 2012, pp. 771-781. [Online]. https://doi.org/10.1109/ICSE.2012.6227141

[2] L. Bilge and T. Dumitraş, "Before We Knew It: an empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, New York, NY, USA, 2012, pp. 833-844. [Online]. https://doi.org/10.1145/2382196.2382284

[3] L. Ablon and A. Bogart, "Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits," RAND Corporation, Mar. 2017. [Online]. Available: https://www.rand.org/content/dam/rand/pubs/research_reports/RR1700/RR1751/RAND_RR1751.pdf

[4] W. Kandek, "The laws of vulnerabilities 2.0," Qualsys, 2009. [Online]. Available: https://www.qualys.com/docs/laws-of-vulnerabilities-2.0.pdf

[5] ISO. "ISO/IEC 27005:2008." [Online]. Available: http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/04/21/42107.html

[6] MITRE, "Common Vulnerabilities and Exposures." [Online]. Available: https://cve.mitre.org/ (accessed Feb. 01, 2020).

[7] J. Erickson, *Hacking: the art of exploitation*, 2nd ed. San Francisco: No starch press, 2008.

[8] Comodo News For Enterprise Security, "Computer Vulnerability | Types of Computer Security Vulnerabilities," Dec. 24, 2018. [Online]. Available: https://enterprise.comodo.com/blog/computer-vulnerability-definition/

[9] Cybersecurity and Infrastructure Security Agency. "Security Tip (ST004-002): Choosing and Protecting Passwords." [Online]. Available: https://www.us-cert.gov/ncas/tips/ST04-002

[10] D. Reichl, "KeePass Password Safe." Accessed February 16, 2020. [Online]. Available: https://keepass.info/

[11] W. Stallings, L. Brown, M. D. Bauer, and A. K. Bhattacharjee, *Computer security: principles and practice*. Pearson Education Upper Saddle River, NJ, USA, 2012.

[12] J. Aycock, *Computer viruses and malware*, vol. 22. Springer Science & Business Media, 2011.

[13] A. Solomon, “A brief history of PC viruses,” *Comput. Fraud Secur. Bull.*, vol. 1993, no. 12, pp. 9-19, 1993.

[14] Portswigger. “What is OS command injection, and how to prevent it?” Accessed December 21, 2019. [Online]. Available: https://portswigger.net/web-security/os-command-injection

[15] Portswigger. “What is SQL Injection? Tutorial & Examples.” Accessed December 22, 2019. [Online]. Available: https://portswigger.net/web-security/sql-injection

[16] T. A. Nidecki, “What Is a Buffer Overflow,” *Acunetix*, Jun. 17, 2019. [Online]. Available: https://www.acunetix.com/blog/web-security-zone/what-is-buffer-overflow/

[17] MITRE. “CWE—CWE-862: Missing Authorization (3.4.1).” [Online]. Available: https://cwe.mitre.org/data/definitions/862.html

[18] MITRE. “CWE—CWE-327: Use of a Broken or Risky Cryptographic Algorithm (3.4.1).” [Online]. Available: https://cwe.mitre.org/data/definitions/327.html

[19] MITRE. “CWE—CWE-306: Missing Authentication for Critical Function (3.4.1).” [Online]. Available: https://cwe.mitre.org/data/definitions/306.html

[20] MITRE. “CWE—CWE-807: Reliance on Untrusted Inputs in a Security Decision (3.4.1).” [Online]. Available: https://cwe.mitre.org/data/definitions/807.html

[21] OWASP. “Command Injection.” Accessed February 06, 2020. [Online]. Available: https://owasp.org/www-community/attacks/Command_Injection

[22] Z. Su and G. Wassermann, “The essence of command injection attacks in web applications,” *ACM SIGPLAN Not.*, vol. 41, no. 1, pp. 372-382, Jan. 2006. [Online]. https://doi.org/10.1145/1111320.1111070

[23] OWASP. “Cross-Site Request Forgery (CSRF).” Accessed December 22, 2019. [Online]. Available: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

[24] OWASP. “Testing for Race Conditions (OWASP-AT-010)..” Accessed December 22, 2019. [Online]. Available: https://www.owasp.org/index.php/Testing_for_Race_Conditions_(OWASP-AT-010)

[25] Web.textfiles.com. Accessed Dec. 22, 2019. [Online]. Available: http://web.textfiles.com/ezines/CRH/crh007.txt

[26] E. Tsyrklevich and B. Yee, “Dynamic detection and prevention of race conditions in file accesses,” In USENIX Security Symposium, Washington, DC, USA, 2003.

[27] T. Farah, R. Shelim, M. Zaman, and D. Alam, “Study of Race Condition: A Privilege Escalation Vulnerability,” *J. Syst. Cybern. Inform.*, vol. 16, no. 1, pp. 22-26, 2018.

[28] ENISA. “Zero-Day.” Accessed December 22, 2019. [Online]. Available: https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/zero-day.

[29] FireEye Security Reimagined, “Zero-Day Danger: A Survey of Zero-Day Attacks and What They Say About the Traditional Security Model.” White Paper, 2015.

[30] P. Wood, B. Nahorney, K. Chandrasekar, S. Wallace, and K. Haley, “Symantec internet security threat report,” *Symantec Corp. Tech Rep*, vol. 21, 2016.

[31] C. Miller, “The Legitimate vulnerability market: the secretive world of 0-day exploit sales.,” presented at the Sixth Workshop on the Economics of Information Security, May 2007.

[32] D. Batchelder et al., “Microsoft Security Intelligence Report, SIRv15,” 2013. [Online]. Available: https://www.microsoft.com/en-us/security/intelligence-report

[33] D. A. D. Smith, “5 examples of zero-day attacks,” *Network World*, Aug. 12, 2013. [Online]. Available: https://www.networkworld.com/article/2168888/5-examples-of-zero-day-attacks.html

[34] L. E. I. Technology, “What Are Zero Day Attacks?,” Lanner, Nov. 02, 2017. [Online]. Available: https://www.lanner-america.com/blog/zero-day-attacks/

[35] M. Hagerott, “Stuxnet and the vital role of critical infrastructure operators and engineers.,” *IJCIP*, vol. 7, no. 4, pp. 244-246, 2014.

[36] N. Falliere, L. O. Murchu, and E. Chien, “W32. Stuxnet dossier,” *White Pap. Symantec Corp Secur. Response*, vol. 5, no. 6, p. 29, 2011.

[37] G. A. Crowther, “National Defense and the Cyber Domain.” [Online]. Available: https://www.heritage.org/military-strength-topical-essays/2019-essays/training-the-foundation-success-combat

[38] J. Pub, “Pub 1-02, Department of Defense Dictionary of Military and Associated Terms,” *Wash. DC Jt. Chiefs Staff*, 1994.

[39] Joint Chiefs of Staff. *Operations*, JP 3-12 (r),” Washington, DC, USA: Joint Chiefs of Staff, 2013.

[40] J. Andress and S. Winterfeld, *Cyber warfare: techniques, tactics and tools for security practitioners*. Elsevier, 2013.

[41] J. R. Lindsay, “Stuxnet and the limits of cyber warfare,” *Secur. Stud.*, vol. 22, no. 3, pp. 365-404, 2013.

[42] N. R. Council, *Technology, policy, law, and ethics regarding U.S. acquisition and use of cyberattack capabilities*. National Academies Press, 2009.

[43] K. C. Laudon and J. P. Laudon, *Management information systems*. Pearson Upper Saddle River, 2015.

[44] M. E. Whitman and H. J. Mattord, *Principles of information security*. Cengage Learning, 2011.

[45] Joint Chiefs of Staff. “US JCS. JP 3-12 (R) Cyberspace Operations.” [Online]. Available: http://www.dtic.mil/doctrine/new_pubs/jp3_12R.pdf

[46] N. Choucri and D. Clark, “Cyberspace and International Relations: Toward an Integrated System,” *Mass. Inst. Technol. Camb. Mass.*, p. 8, 2011.

[47] D. Raymond, T. Cross, G. Conti, and M. Nowatkowski, “Key terrain in cyberspace: Seeking the high ground,” in *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, 2014, pp. 287-300.

[48] L. A. Johnson, K. L. Dempsey, R. S. Ross, S. Gupta, and D. Bailey, “SP 800-128. Guide for security-focused configuration management of information systems,” 2011.

[49] K. L. Dempsey, G. A. Witte, and D. Rike, “Summary of NIST SP 800-53, Revision 4: Security and Privacy Controls for Federal Information Systems and Organizations,” 2014.

[50] ISO, “ISO/IEC 27005:2018,” Apr. 21, 2017. [Online]. Available: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/52/75281.html

[51] G. B. White, E. A. Fisch, and U. W. Pooch, *Computer system and network security*, vol. 7. CRC press, 1995.

[52] M. A. Talib, M. El Barachi, A. Khelifi, and O. Ormandjieva, “Guide to ISO 27001: UAE case study,” *Issues Informing Sci. Inf. Technol.*, vol. 7, pp. 331-349, 2012.

[53] M. Nieles, K. Dempsey, and V. Pillitteri, “An Introduction to Information Security,” National Institute of Standards and Technology, NIST Special Publication (SP) 800-12 Rev. 1, Jun. 2017. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-12/rev-1/final

[54] S. Mohurle and M. Patil, “A brief study of wannacry threat: Ransomware attack 2017,” *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, 2017.

[55] CVE. “CVE-2017-0143 : The SMBv1 server in Microsoft Windows Vista SP2; Windows Server 2008 SP2 and R2 SP1; Windows 7 SP1; Windows 8.1; Windows.” [Online]. Available: https://www.cvedetails.com/cve/CVE-2017-0143/

[56] Rapid7. “DoublePulsar Explained,” Accessed May 14, 2020. [Online]. Available: https://www.rapid7.com/security-response/doublepulsar/

[57] K. A. O. Da-Yu, S.-C. HSIAO, and T. S. O. Raylin, “Analyzing WannaCry Ransomware Considering the Weapons and Exploits,” in *2019 21st International Conference on Advanced Communication Technology (ICACT)*, 2019, pp. 1098-1107.

[58] FireEye. “SMB Exploited: WannaCry Use of ‘EternalBlue,’” [Online]. Available: https://www.fireeye.com/blog/threat-research/2017/05/smb-exploited-wannacry-use-of-eternalblue.html

[59] Brandon, “Mssecvc.exe/Taskche.exe Virus Removal (Wannacry),” Virus Removal Guides, May 13, 2017. [Online]. Available: https://howtoremove.guide/mssecvc-exe-taskche-exe-virus-remove/

[60] Dyn. Bus., “There are two types of companies: Those who know they’ve been hacked & those who don’t.” [Online]. Available: https://dynamicbusiness.com.au/topics/technology/there-are-two-types-of-companies-those-who-know-theyve-been-hacked-those-who-dont.html

[61] S. Frei, B. Tellenbach, and B. Plattner, “0-day patch exposing vendors (in) security performance,” *BlackHat Eur.*, 2008.

[62] SANS Institute. “Reading Room—Security Basics.” [Online]. Available: https://www.sans.org/reading-room/whitepapers/basics/paper/525

[63] Project Zero. “0day ‘In the Wild,’” Proj. Zero, [[Online]. Available: https://googleprojectzero.blogspot.com/p/0day.html

[64] W. He, H. Li, and J. Li, “Unknown Vulnerability Risk Assessment Based on Directed Graph Models: A Survey,” *IEEE Access*, vol. 7, pp. 168201-168225, 2019.

[65] L. Wang, M. Zhang, S. Jajodia, A. Singhal, and M. Albanese, “Modeling network diversity for evaluating the robustness of networks against zero-day attacks,” in *European Symposium on Research in Computer Security*, 2014, pp. 494-511.

[66] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, “On the effectiveness of address-space randomization,” in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 298-307.

[67] O. Tytarenko, “Selection of the best security controls for rapid development of enterprise-level cyber security,” M.S. thesis, Dept. of CS, NPS, Monterey, CA, USA, 2017.

[68] C. C. Wright, “Host intrusion prevention system using software and user behavior analysis,” U.S. *Pat. 8607340*, Dec. 2013, [Online]. Available: https://patentimages.storage.googleapis.com/ae/32/ca/af7ed96106b5b5/US8607340.pdf

[69] U. K. Singh, C. Joshi, and D. Kanellopoulos, “A framework for zero-day vulnerabilities detection and prioritization,” *J. Inf. Secur. Appl.*, vol. 46, pp. 164-172, 2019.

[70] T. Nadean H., *Cybersecurity Blue Team Toolkit*. Indianapolis, IN, USA: Wiley, 2019.

[71] Decipher. “Project Zero Wants You To Help Make 0-Day Hard.” [Online]. Available: https://duo.com/decipher/project-zero-wants-you-to-help-make-0-day-hard

[72] J. Jacobi, “From Zero to Zero Day.” Media. [Online]. Available: https://media.ccc.de/v/35c3-9657-from_zero_to_zero_day#t=209

[73] D. Warren, “Vulnerability Discovery.” CMU. 2016, [Online]. Available: https://resources.sei.cmu.edu/asset_files/Presentation/2016_017_001_474262.pdf

[74] S. M. Rajasooriya, C. P. Tsokos, and P. K. K. H. Kaluarachchilage, “Vulnerability life cycle exploitation timing modeling,” May 12, 2020.

[75] V. Sharma, J. D. Lim, J. N. Kim, and I. You, “Saca: Self-aware communication architecture for IOT using mobile fog servers,” *Mob. Inf. Syst.*, vol. 2017, 2017.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California