ECE 4309

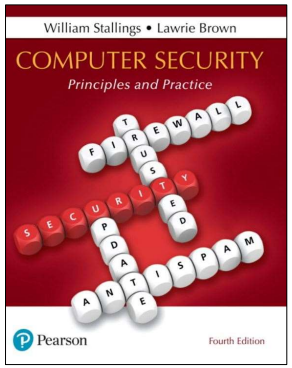
Malwares – part 1

Dr. Valerio Formicola

CalPolyPomona

Computer Security: Principles and Practice

Fourth Edition



Chapter 6

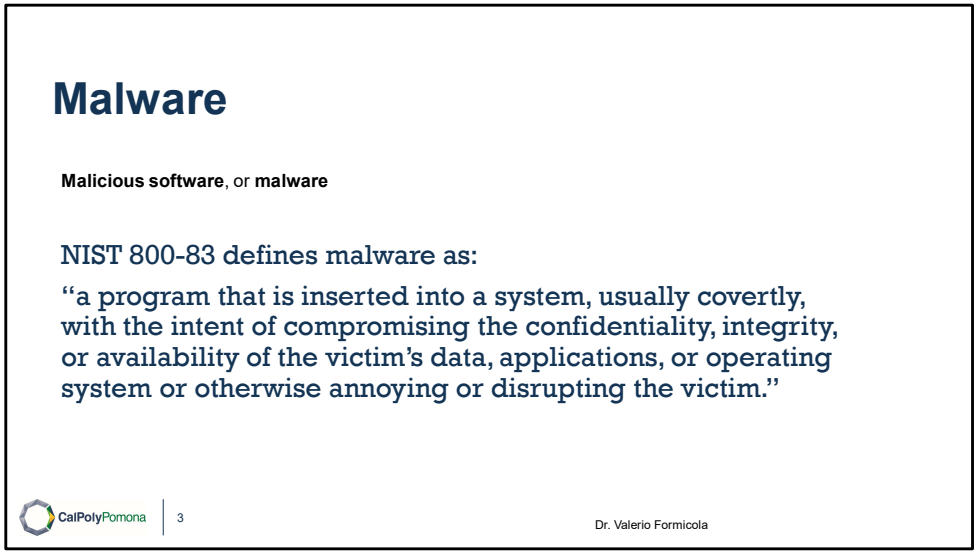
Malicious Software

Pearson

If this PowerPoint presentation contains mathematical equations, you may need to check that your computer has the following installed:
1) MathType Plugin
2) Math Player (free versions available)
3) NVDA Reader (free versions available)

Slides in this presentation contain hyperlinks. JAWS users should be able to get a list of links by using INSERT+F7

This chapter examines the wide spectrum of malware threats and countermeasures. We begin with a survey of various types of malware, and offer a broad classification based first on the means malware uses to spread or **propagate**, and then on the variety of actions or **payloads** used once the malware has reached a target. Propagation mechanisms include those used by viruses, worms, and Trojans. Payloads include system corruption, bots, phishing, spyware, and rootkits. The discussion concludes with a review of countermeasure approaches.



Malicious software, or **malware**, arguably constitutes one of the most significant categories of threats to computer systems. NIST SP 800-83 *(Guide to Malware Incident Prevention and Handling for Desktops and Laptops,* July 2013) defines malware as "a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system or otherwise annoying or disrupting the victim." Hence, we are concerned with the threat malware poses to application programs, to utility programs, such as editors and compilers, and to kernel-level programs. We are also concerned with its use on compromised or malicious Web sites and servers, or in especially crafted spam e-mails or other messages, which aim to trick users into revealing sensitive personal information.

Table 6.1 Malware Terminology (1 of 3)

Name	Description
Advanced Persistent Threat (APT)	Cybercrime directed at business and political targets, using a wide variety of intrusion technologies and malware, applied persistently and effectively to specific targets over an extended period, often attributed to state-sponsored organizations.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.
Attack kit	Set of tools for generating new malware automatically using a variety of supplied propagation and payload mechanisms.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Backdoor (trapdoor)	Any mechanism that bypasses a normal security check; it may allow unauthorized access to functionality in a program, or onto a compromised system.
Downloaders	Code that installs other items on a machine that is under attack. It is normally included in the malware code first inserted on to a compromised system to then import a larger malware package.
Drive-by-download	An attack using code on a compromised website that exploits a browser Vulnerability to attack a client system when the site is viewed.
Exploits	Code specific to a single vulnerability or set of vulnerabilities.

The terminology in this area presents problems because of a lack of universal agreement on all of the terms and because some of the categories overlap. Table 6.1 is a useful guide to some of the terms in use.

Table 6.1 Malware Terminology (2 of 3)

Name	Description
Flooders (DoS client)	Used to generate a large volume of data to attack networked computer systems, by carrying out some form of denial-of-service (DoS) attack.
Keyloggers	Captures keystrokes on a compromised system.
Logic bomb	Code inserted into malware by an intruder. A logic bomb lies dormant until a Predefined condition is met; the code then triggers some payload.
Macro virus	A type of virus that uses macro or scripting code, typically embedded in a Document or document template, and triggered when the document is viewed or edited, to run and replicate itself into other such documents.
Mobile code	Software (e.g., script and macro) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access.
Spammer programs	Used to send large volumes of unwanted e-mail.
Spyware	Software that collects information from a computer and transmits it to another system by monitoring keystrokes, screen data, and/or network traffic; or by scanning files on the system for sensitive information.

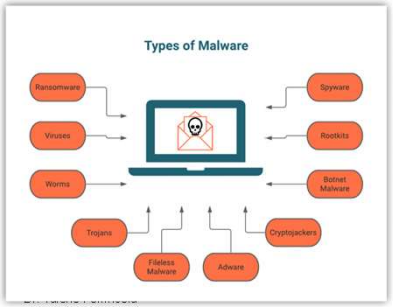
Table 6.1 Malware Terminology (3 of 3)

Name	Description
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes it.
Virus	Malware that, when executed, tries to replicate itself into other executable machine or script code; when it succeeds, the code is said to be infected. When the infected code is executed, the virus also executes.
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network, by exploiting software vulnerabilities in the target system, or using captured authorization credentials.
Zombie, bot	Program installed on an infected machine that is activated to launch attacks on other machines.

Classification of Malware

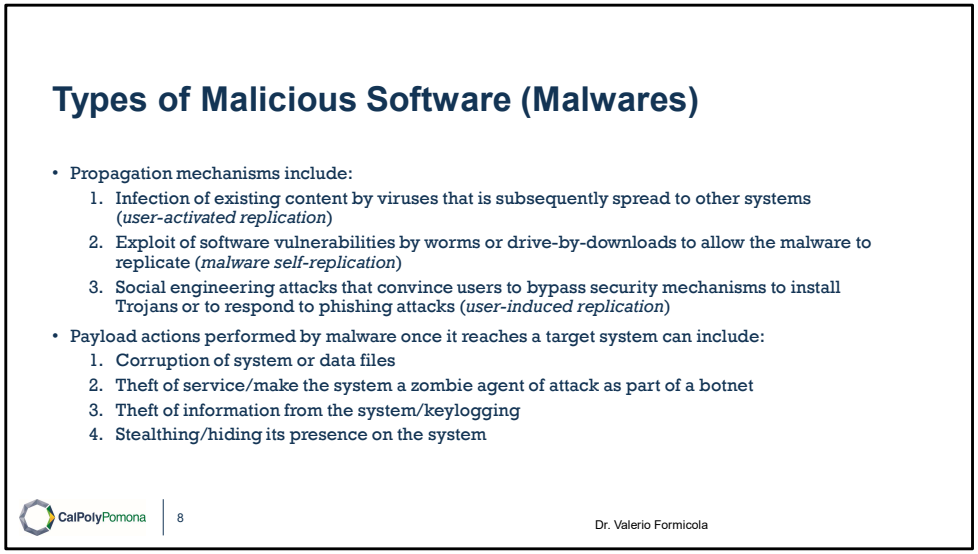
- Categories used for classification:
  - **Propagation Method:** Based first on how it spreads or propagates to reach the desired targets
  - **Actions on the Target:** Actions or payloads it performs once a target is reached

Additional criteria for classification:
- The need for a host program (parasitic code such as viruses)
- Independent, self-contained programs (worms, trojans, and bots)
- Malware that does not replicate (trojans and spam e-mail)
- Malware that does replicate (viruses and worms)



A number of authors attempt to classify malware, as shown in the survey and proposal of [HANS04]. Although a range of aspects can be used, one useful approach classifies malware into two broad categories, based first on how it spreads or propagates to reach the desired targets; and then on the actions or payloads it performs once a target is reached.

Earlier approaches to malware classification distinguished between those that need a host program, being parasitic code such as viruses, and those that are independent, self-contained programs run on the system such as worms, trojans, and bots. Another distinction used was between malware that does not replicate, such as trojans and spam e-mail, and malware that does, including viruses and worms.



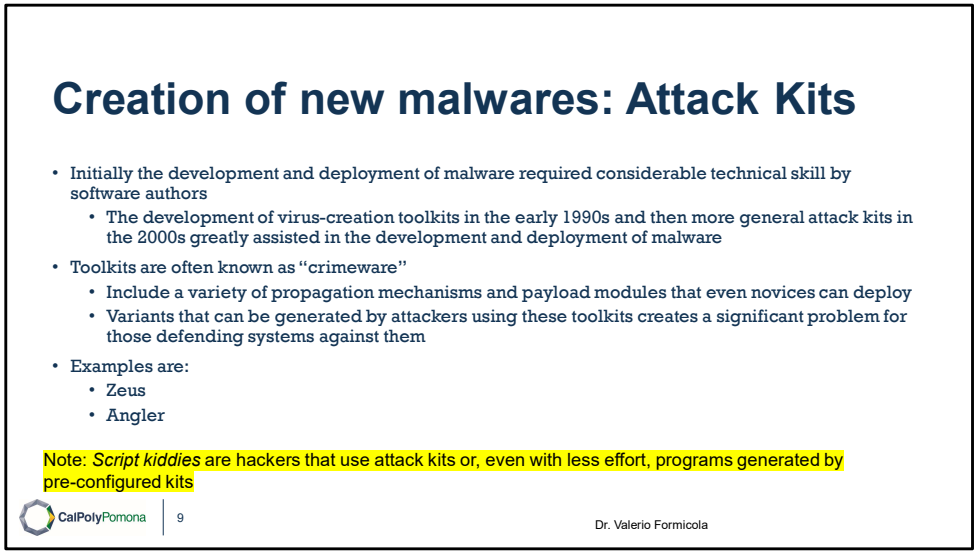
Propagation mechanisms include infection of existing executable or interpreted content by viruses that is subsequently spread to other systems; exploit of software vulnerabilities either locally or over a network by worms or drive-by-downloads to allow the malware to replicate; and social engineering attacks that convince users to bypass security mechanisms to install Trojans, or to respond to phishing attacks.

Payload actions performed by malware once it reaches a target system can include corruption of system or data files; theft of service in order to make the system a zombie agent of attack as part of a botnet; theft of information from the system, especially of logins, passwords, or other personal details by keylogging or spyware programs; and stealthing where the malware hides its presence on the system from attempts to detect and block it.

While early malware tended to use a single means of propagation to deliver a single payload, as it evolved, we see a growth of blended malware that incorporates a range of both propagation mechanisms and payloads that increase its ability to spread, hide, and perform a range of actions on targets. A **blended attack** uses multiple methods of infection or propagation, to maximize the speed of contagion

and the severity of the attack. Some malware even support an update mechanism that allows it to change the range of propagation and payload mechanisms utilized once it is deployed.

In the following sections, we survey these various categories of malware, and then follow with a discussion of appropriate countermeasures.



Initially, the development and deployment of malware required considerable technical skill by software authors. This changed with the development of virus- creation toolkits in the early 1990s, and then later of more general attack kits in the 2000s, that greatly assisted in the development and deployment of malware [FOSS10]. These toolkits, often known as **crimeware**, now include a variety of propagation mechanisms and payload modules that even novices can combine, select, and deploy. They can also easily be customized with the latest discovered vulnerabilities in order to exploit the window of opportunity between the publication of a weakness and the widespread deployment of patches to close it. These kits greatly enlarged the population of attackers able to deploy malware. Although the malware created with such toolkits tends to be less sophisticated than that designed from scratch, the sheer number of new variants that can be generated by attackers using these toolkits creates a significant problem for those defending systems against them.

The Zeus crimeware toolkit is a prominent example of such an attack kit, which was used to generate a wide range of very effective, stealthed malware that facilitates a range of criminal activities, in particular capturing and exploiting banking credentials [BINS10]. The Angler exploit kit, first seen in 2013, was the most active kit seen in 2015, often distributed via malvertising that exploited Flash vulnerabilities. It is sophisticated and

technically advanced, in both attacks executed and counter-measures deployed to resist detection. There are a number of other attack kits in active use, though the specific kits change from year to year as attackers continue to evolve and improve them [SYMA16].

Diffusion of malwares

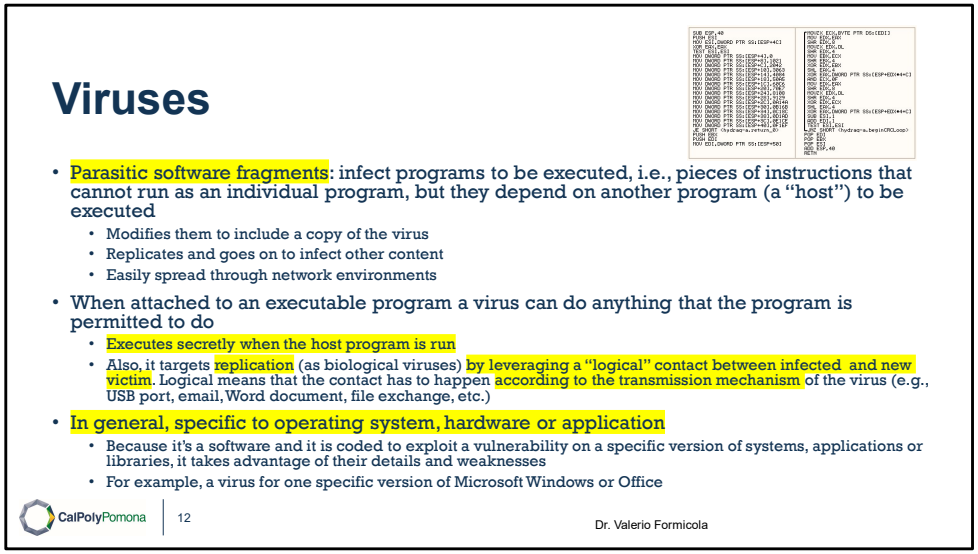
- As more capable and financially funded hackers exist, more and more actors can generate malware
  - Remember: Cyber-Criminals, State-sponsored, hobbyists, Organizations that sell their services to companies and nations

- This has significantly changed the resources available and motivation behind the rise of malware and has led to development of a large underground economy involving the sale of attack kits, access to compromised hosts, and to stolen information

Dr. Valerio Formicola

Another significant malware development over the last couple of decades is the change from attackers being individuals, often motivated to demonstrate their technical competence to their peers, to more organized and dangerous attack sources. These include politically motivated attackers, criminals, and organized crime; organizations that sell their services to companies and nations, and national government agencies, as we discuss in Section 8.1. This has significantly changed the resources available and motivation behind the rise of malware, and indeed has led to development of a large underground economy involving the sale of attack kits, access to compromised hosts, and to stolen information.


Viruses
CalPolyPomona



The first category of malware propagation concerns parasitic software fragments that attach themselves to some existing executable content. The fragment may be machine code that infects some existing application, utility, or system program, or even the code used to boot a computer system. Computer virus infections formed the majority of malware seen in the early personal computer era. The term "computer virus" is still often used to refer to malware in general, rather than just computer viruses specifically. More recently, the virus software fragment has been some form of scripting code, typically used to support active content within data files such as Microsoft Word documents, Excel spreadsheets, or Adobe PDF documents.

A computer virus is a piece of software that can "infect" other programs, or indeed any type of executable content, by modifying them. The modification includes injecting the original code with a routine to make copies of the virus code, which can then go on to infect other content. Computer viruses first appeared in the early 1980s, and the term itself is attributed to Fred Cohen. Cohen is the author of a groundbreaking book on the subject [COHE94]. The Brain virus, first seen in 1986, was one of the first to target MSDOS systems, and resulted in a significant number of infections for this time.

Biological viruses are tiny scraps of genetic code—DNA or RNA—that can take over the machinery of a living cell and trick it into making thousands of flawless replicas of the original virus. Like its biological counterpart, a computer virus carries in its instructional code the recipe for making perfect copies of itself. The typical virus becomes embedded in a program, or carrier of executable content, on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of code, a fresh copy of the virus passes into the new location.
Thus, the infection can spread from computer to computer, aided by unsuspecting users, who exchange these programs or carrier files on disk or USB stick; or who send them to one another over a network. In a network environment, the ability to access documents, applications, and system services on other computers provides a perfect culture for the spread of such viral code.

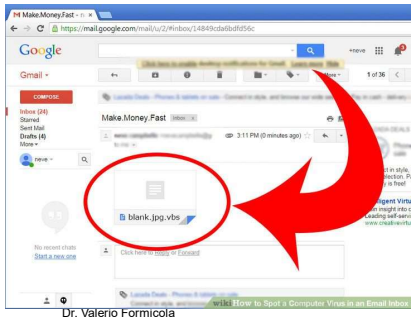
A virus that attaches to an executable program can do anything that the program is permitted to do. It executes secretly when the host program is run. Once the virus code is executing, it can perform any function, such as erasing files and programs, that is allowed by the privileges of the current user. One reason viruses dominated the malware scene in earlier years was the lack of user authentication and access controls on personal computer systems at that time. This enabled a virus to infect any executable content on the system. The significant quantity of programs shared on floppy disk also enabled its easy, if somewhat slow, spread. The inclusion of tighter access controls on modern operating systems significantly hinders the ease of infection of such traditional, machine executable code, viruses. This resulted in the development of macro viruses that exploit the active content supported by some documents types, such as Microsoft Word or Excel files, or Adobe PDF documents. Such documents are easily modified and shared by users as part of their normal system use, and are not protected by the same access controls as programs. Currently, a viral mode of infection is typically one of several propagation mechanisms
used by contemporary malware, which may also include worm and Trojan capabilities.

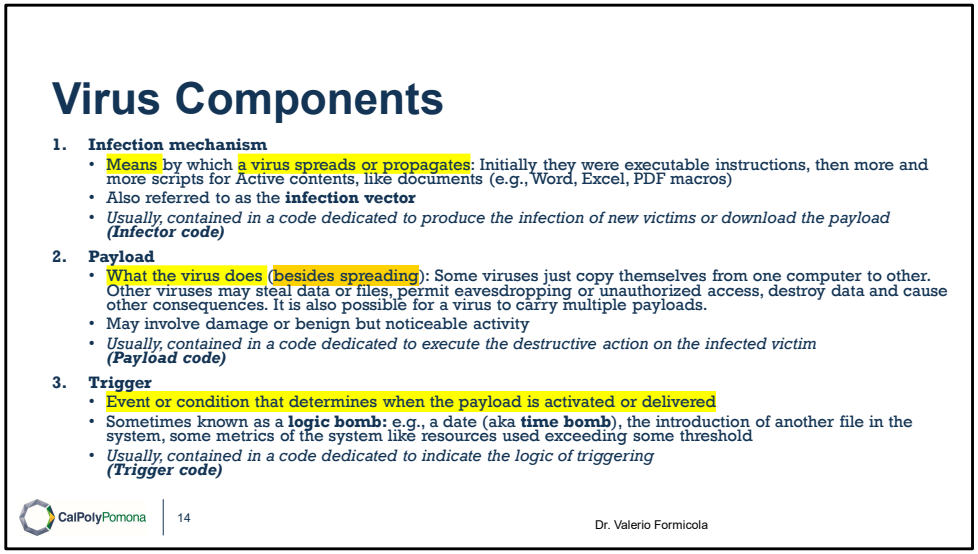
Main characteristic of a virus: user-based activation required

- A computer virus lives within a host, such as a document or executable file, and **requires human interaction to spread.**

Example:
You receive an email (that you're not expecting) with an intriguing (clickbait) title like "Made some changes — please check." Attached to the email is a file with a name like "Updates" — it may be a DOC or EXE file.

- If it's a DOC file, once you download it, you'll be prompted to enable macros (programmed rules that help simplify repetitive tasks). This action triggers the virus.
- If the file is an EXE, downloading it and running it triggers the virus. The virus will then commandeer your computer's resources to copy itself and spread, damaging your devices and files or stealing your personal data.
- If it's an advertisement (*clickbait*), you might click attracted by some clickbait and be redirected to an infected website.





[AYCO06] states that a computer virus has three parts. More generally, many contemporary types of malware also include one or more variants of each of these components:

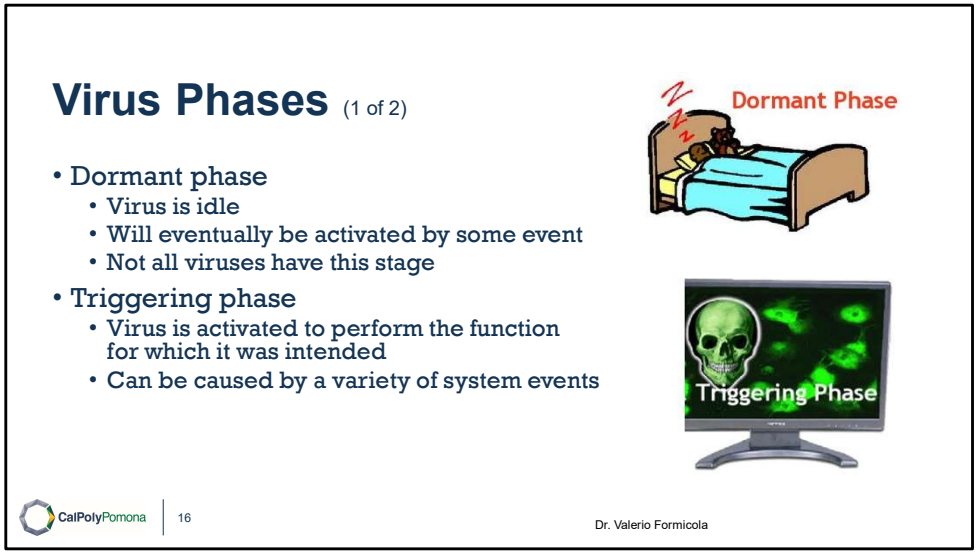
• **Infection mechanism** : The means by which a virus spreads or propagates, enabling it to replicate. The mechanism is also referred to as the **infection vector**.

• **Trigger**: The event or condition that determines when the payload is activated or delivered, sometimes known as a **logic bomb.**

• **Payload**: What the virus does, besides spreading. The payload may involve damage or may involve benign but noticeable activity.

Examples of actions done by virus payloads (i.e., impact)

- **Data theft:** Particularly common is the theft of sensitive information such as login credentials or financial information through various forms of data breaches.
- **Activity monitoring:** An executed malicious payload may serve to monitor user activity on a computer, this can be done for the purposes of spying, blackmail, or to aggregate consumer behavior which can be sold to advertisers.
- **Displaying advertisements:** Some malicious payloads work to display persistent, unwanted ads such as pop-ups and pop-unders to the victim.
- **Deleting or modifying files**: This is one of the most serious consequences to arise from a malicious payload. Files can be deleted or modified to either affect the behavior of a computer, or even disable the operating system and/or startup processes. For example, some malicious payloads are designed to 'brick' smartphones, meaning they can no longer be turned on or used in any way.
- **Downloading new files:** Some malicious payloads come in very lightweight files that are easy to distribute, but once executed they will trigger the download of a much larger piece of malicious software.
- **Running background processes:** A malicious payload can also be triggered to quietly run processes in the background, such as cryptocurrency mining or data storage.
- **Enable backdoor access:** from externals to a user's computer



During its lifetime, a typical virus goes through the following four phases:

• **Dormant phase**: The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.

• **Propagation phase:** The virus places a copy of itself into other programs or into certain system areas on the disk. The copy may not be identical to the propagating version; viruses often morph to evade detection. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.

• **Triggering phase**: The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

• **Execution phase**: The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

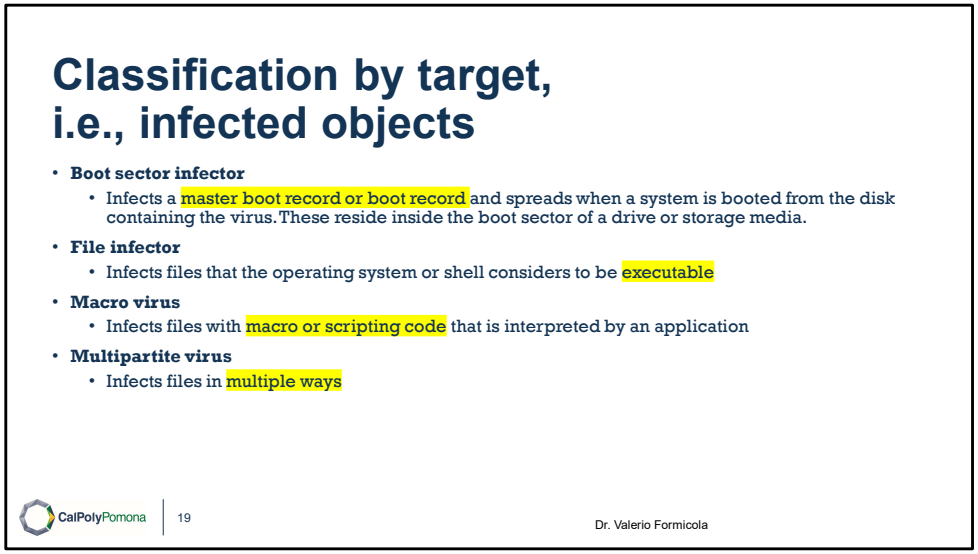
Most viruses that infect executable program files carry out their work in a manner that is specific to a particular operating system and, in some cases, specific to a particular hardware platform. Thus, they are designed to take advantage of the details and weaknesses of particular systems. Macro viruses though, target specific document types, which are often supported on a variety of systems.

Virus Phases (2 of 2)

- Propagation phase
  - Virus places a copy of itself into other programs or into certain system areas on the disk
  - May not be identical to the propagating version
  - Each infected program will now contain a clone of the virus which will itself enter a propagation phase
- Execution phase
  - Function is performed
  - May be harmless or damaging

Virus Classifications

There is no unique model,
but most commonly the attributes
are like in the figure



There has been a continuous arms race between virus writers and writers of anti- virus software since viruses first appeared. As effective countermeasures are developed for existing types of viruses, newer types are developed. There is no simple or universally agreed upon classification scheme for viruses. In this section, we follow [AYCO06] and classify viruses along two orthogonal axes: the type of target the virus tries to infect and the method the virus uses to conceal itself from detection by users and anti-virus software.

A virus classification by target includes the following categories:

• **Boot sector infector**: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.

• **File infector:** Infects files that the operating system or shell consider to be executable.

• **Macro virus**: Infects files with macro or scripting code that is interpreted by an application.

• **Multipartite virus**: Infects files in multiple ways. Typically, the multipartite virus is capable of infecting multiple types of files, so that virus eradication must deal with all of the possible sites of infection.

Classification by hiding strategy (camouflage)

- **Encrypted virus**
  - A portion of the virus creates a random encryption key and encrypts the remainder of the virus
  - Key is changed in each replication, so it's hard to detect using regular bit sequences
- **Stealth virus**
  - A form of virus explicitly designed to hide itself from detection by anti-virus software
- **Oligomorphic virus**
  - Oligomorphism is an advanced form of the encryption. It contains a collection of different decryptors, which are randomly chosen for a new victim. In such a way, the decryptor code is not identical in various instances. T
- **Polymorphic virus**
  - A virus that mutates with every infection. For example, add random instructions between more targeted instructions (i.e., the instructions that replicate or execute the payloads)
- **Metamorphic virus**
  - A virus that mutates and rewrites itself completely at each iteration and may cha appearance

Note: some viruses might use a combination of the techniques above to hide

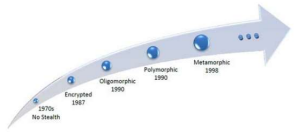


Figure 1: Evolution timeline of Camouflage techniques appearance in malware

A virus classification by concealment strategy includes the following categories:

• **Encrypted virus**: A form of virus that uses encryption to obscure its content. A portion of the virus creates a random encryption key and encrypts the remainder of the virus. The key is stored with the virus. When an infected program is invoked, the virus uses the stored random key to decrypt the virus.
When the virus replicates, a different random key is selected. Because the bulk of the virus is encrypted with a different key for each instance, there is no constant bit pattern to observe.

• **Stealth virus** : A form of virus explicitly designed to hide itself from detection by anti-virus software. Thus, the entire virus, not just a payload is hidden. It may use code mutation, compression, or rootkit techniques to achieve this.

• **Polymorphic virus**: A form of virus that creates copies during replication that are functionally equivalent but have distinctly different bit patterns, in order to defeat programs that scan for viruses. In this case, the "signature" of the virus will vary with each copy. To achieve this variation, the virus may randomly insert

superfluous instructions or interchange the order of independent instructions. A more effective approach is to use encryption. The strategy of the encryption virus is followed. The portion of the virus that is responsible for generating keys and performing encryption/decryption is referred to as the mutation engine . The mutation engine itself is altered with each use.

• **Metamorphic virus**: As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, using multiple transformation techniques, increasing the difficulty of detection. Metamorphic viruses may change their behavior as well as their appearance.

Encrypted Virus

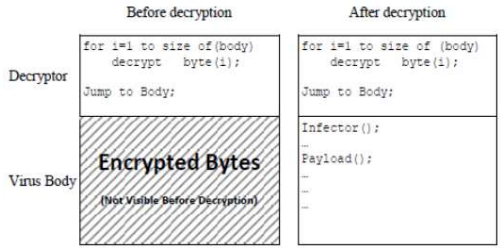


Figure 2: Structure of encrypted virus

Rad, Babak Bashari, Maslin Masrom, and Suhaimi Ibrahim. "Camouflage in malware: from encryption to metamorphism." *International Journal of Computer Science and Network Security* 12, no. 8 (2012): 74-83.

Oligomorphic virus

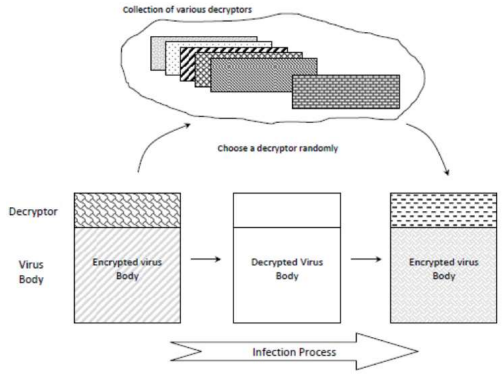
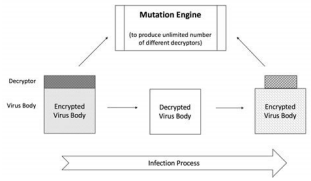
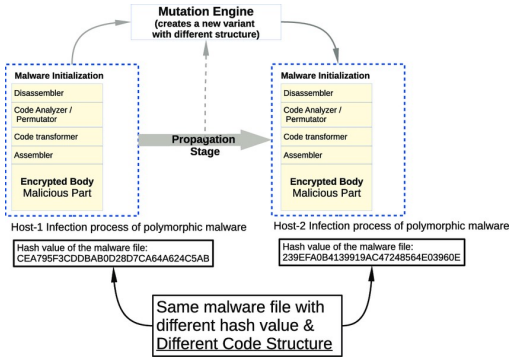


Figure 3: Structure and mechanism of oligomorphic virus

Polymorphic virus





Dr. Valerio Formicola

Metamorphic virus

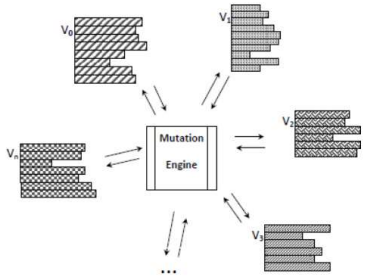


Figure 5: Metamorphic virus propagation scheme

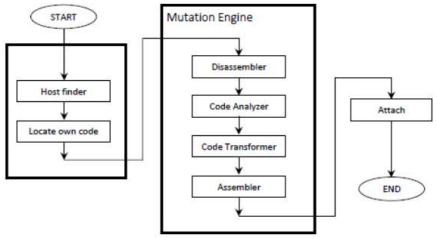
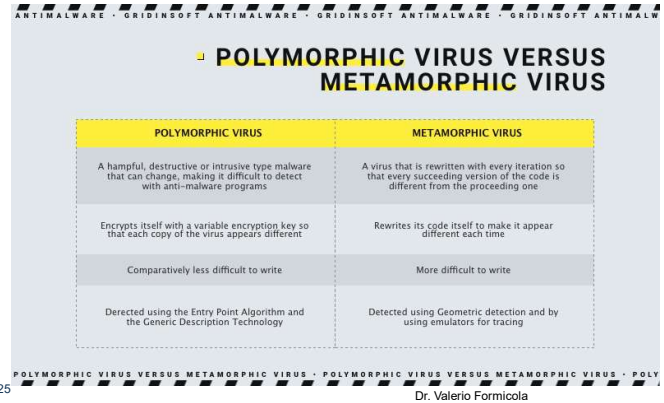


Figure 6: Structure of replicator and mutation engine in metamorphic virus

Metamorphic virus



Stealth virus

- A computer virus that avoids detection by hiding itself after infecting the machine. The stealth virus may hide in different locations such as **boot sectors (e.g., Master Boot Sector MBR),** various files, and **undetectable computer areas**, effectively tricking anti-virus software
- In order to avoid detection, stealth viruses also self-modify in the following ways:
  - Code Modification: The stealth virus changes the code and virus signature of each infected file.
  - Encryption: The stealth virus encrypts data via simple encryption and uses a different encryption key for each infected file.

Table 3: Two versions of W32.Evol

Binary Code Sequence	Assembly Code	
C7060F000055	mov	[esi], 5500000Fh
C746048BEC5151	mov	[esi+0004],5151EC8Bh
String Signature:	C7060F000055C746048BEC5151	

Binary Code Sequence	Assembly Code	
BF0F000055	mov	edi,5500000Fh
893E	mov	[esi],edi
5F	pop	edi
52	push	edx
B640	mov	dh,40
BA8BEC5151	mov	edx,5151EC8Bh
53	push	ebx
8BDA	mov	ebx,edx
895E04	mov	[esi+0004],ebx
String Signature:	BF0F000055893E5F52B640BA8BEC515153 8BDA895E04	

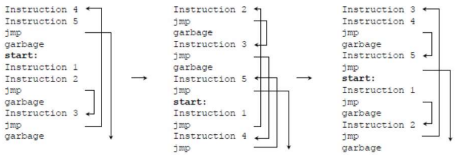
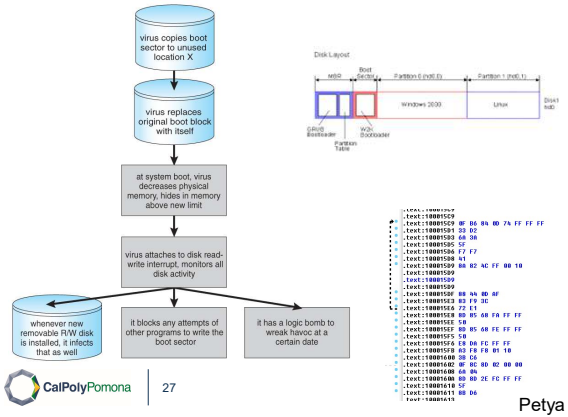
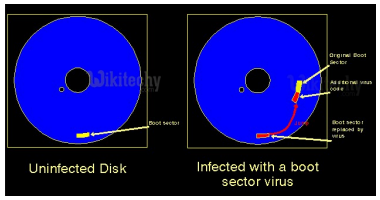
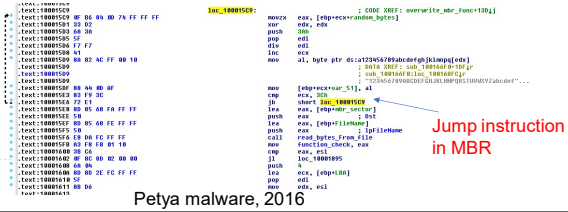


Figure 7: Code transposition in Zperm virus

Example: infecting and hiding in the boot sector of disk





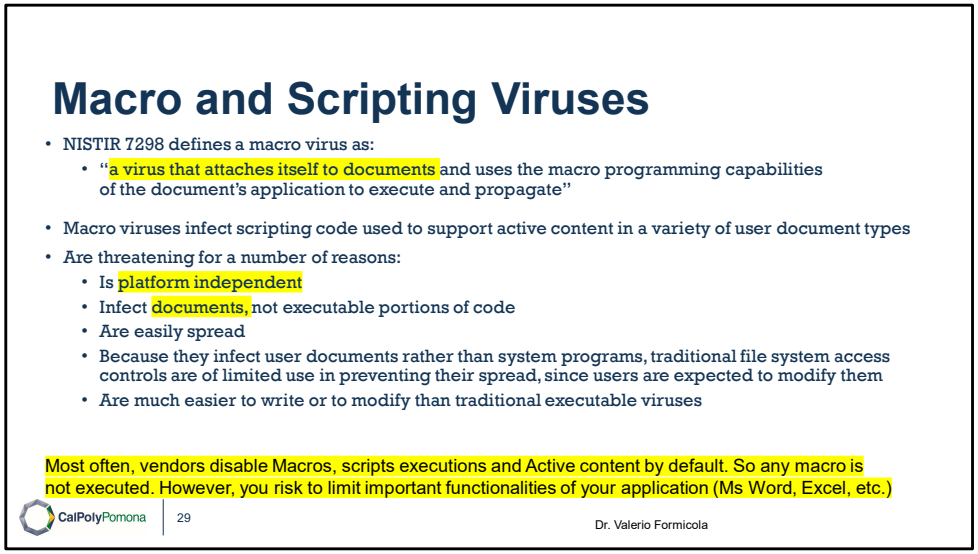


CalPolyPomona | 27

Petya malware, 2016

https://shasaurabh.blogspot.com/2017/07/notpetyapetya-overwriting-system-mbr.html


Macro and scripts
viruses
CalPolyPomona



In the mid-1990s, macro or scripting code viruses became by far the most prevalent type of virus. NISTIR 7298 (*Glossary of Key Information Security Terms,* May 2013) defines a **macro virus** as a virus that attaches itself to documents and uses the macro programming capabilities of the document's application to execute and propagate. Macro viruses infect scripting code used to support active content in a variety of user document types. Macro viruses are particularly threatening for a number of reasons:

1. A macro virus is platform independent. Many macro viruses infect active content in commonly used applications, such as macros in Microsoft Word documents or other Microsoft Office documents, or scripting code in Adobe PDF documents. Any hardware platform and operating system that supports these applications can be infected.

2. Macro viruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of documents rather than programs.

3. Macro viruses are easily spread, as the documents they exploit are shared in normal use. A very common

method is by electronic mail, particularly since these documents can sometimes be opened automatically without prompting the user.

4. Because macro viruses infect user documents rather than system programs, traditional file system access controls are of limited use in preventing their spread, since users are expected to modify them.

5. Macro viruses are much easier to write or to modify than traditional executable viruses.

Macro viruses take advantage of support for active content using a scripting or macro language, embedded in a word processing document or other type of file. Typically, users employ macros to automate repetitive tasks and thereby save keystrokes. They are also used to support dynamic content, form validation, and other useful tasks associated with these documents.

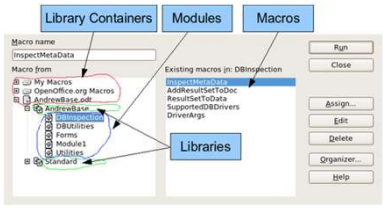
Microsoft Word and Excel documents are common targets due to their widespread use. Successive releases of MS Office products provide increased protection against macro viruses. For example, Microsoft offers an optional Macro Virus Protection tool that detects suspicious Word files and alerts the customer to the potential risk of opening a file with macros. Office 2000 improved macro security by allowing macros to be digitally signed by their author, and for authors to be listed as trusted. Users were then warned if a document being opened contained unsigned, or signed but untrusted, macros, and were advised to disable macros in this case. Various antivirus product vendors have also developed tools to detect and remove macro viruses. As in other types of malware, the arms race continues in the field of macro viruses, but they no longer are the predominant malware threat.

Another possible host for macro virus–style malware is in Adobe's PDF documents. These can support a range of embedded components, including Javascript and other types of scripting code. Although recent PDF viewers include measures to warn users when such code is run, the message the user is shown can be manipulated to trick them into permitting its execution. If this occurs, the code could potentially act as a virus to infect other PDF documents the user can access on their system. Alternatively, it can install a Trojan, or act as a worm, as we discuss later [STEV11].

Why MACROs and scripts as a virus

- Viruses dominated the malware scene in earlier years due to lack of user authentication and access controls on personal computer systems at that time.
  - This enabled a virus to infect any executable content on the system.
  - The significant quantity of programs shared on floppy disk also enabled its easy, if somewhat slow, spread.
- The inclusion of tighter access controls on modern operating systems significantly hinders the ease of infection of such traditional, machine executable code, viruses. This resulted in the development of macro viruses that exploit the active content supported by some documents types, such as Microsoft Word or Excel files, or Adobe PDF documents. Such documents are easily modified and shared by users as part of their normal system use, and are not protected by the same access controls as programs.

Example: a macro




```
REM  *****  BASIC  *****

Sub send_email(sAttachmentURL as string, sEmailAddress as string, sSubject as string, sMessageBody as string)

 Dim eMailer as Object
 Dim eMailClient as Object
 Dim eMessage as Object

        eMailer = createUnoService( "com.sun.star.system.SimpleSystemMail" )
        eMailClient = eMailer.querySimpleMailClient()
        eMessage = eMailClient.createSimpleMailMessage()
                eMessage.setRecipient(sMailAddress)
                eMessage.setSubject(sSubject)
        eMailClient.sendSimpleMailMessage( eMessage, 0 ) 'if you want to handle the sending manually in the mail client software
'       eMailClient.sendSimpleMailMessage ( eMessage,com.sun.star.system.SimpleMailClientFlags.NO_USER_INTERFACE ) 'Silent sendi
End Sub
```

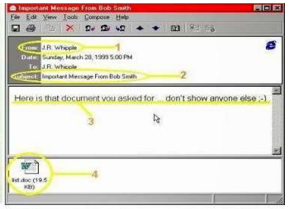
OpenOffice.org macro

- With a MACRO you can: open files, create new files, delete files, execute media files, send emails, etc.
- The depend on the application interpreting the macro (i.e., scripts for Ms Office, OpenOffice, Adobe PDF, …)
- Executed at the moment indicated in the macro: e.g., when you open a document, create a new document, close a document, edit the document, etc.

Example: Melissa virus

- *Date of Attack* – **March 26, 1999**
- *Attacker* – **30 year old David Smith**
- *Victims* – **thousands of Microsoft Word 97 and Word 2000 email users**
- *Damage* - **$80 million**

David L. Smith



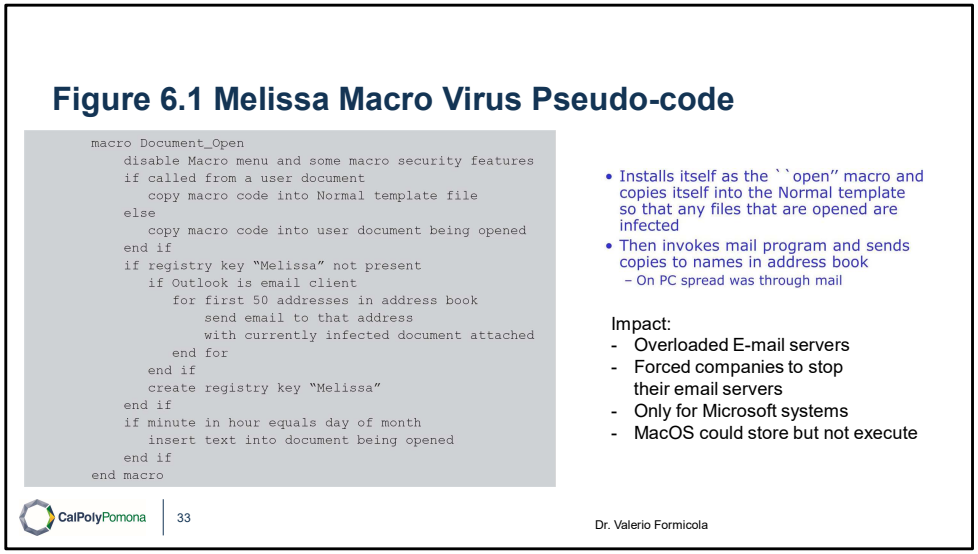
This is what you'll see if you receive the Melissa Virus

1. It may be from someone you know.
2. The subject line will read, "Important Message From" and the name may be someone you know.
3. The body of the mail will read, "Here is that document you asked for don't show anyone else ;-)"
4. The attachment, where this nasty little bug lives, has the name of list.doc.

It has characteristics of Viruses, Worms and Trojans

The Virus part is because of users required to **open the document attached to emails**
(following versions just needed to open the email)

Dr. Valerio Formicola



Although macro languages may have a similar syntax, the details depend on the application interpreting the macro, and so will always target
documents for a specific application. For example, a Microsoft Word macro, including a macro virus, will be different to an Excel macro. Macros can either be saved with a document, or be saved in a global template or worksheet. Some macros are run automatically when certain actions occur. In Microsoft Word, for example, macros can run when Word starts, a document is opened, a new document is created, or when a document is closed. Macros can perform a wide range of operations, not just only on the document content, but can read and write files, and call other applications.

As an example of the operation of a macro virus, pseudo-code for the Melissa macro virus is shown in Figure 6.1. This was a component of the Melissa e-mail worm that we will describe further in the next section. This code would be introduced onto a system by opening an infected Word document, most likely sent by e-mail. This macro code is contained in the Document_Open macro, which is automatically run when the document is opened. It first disables the Macro menu and some related security features, making it harder for the user stop or remove its operation. Next it checks to see if it is being run from an infected document, and if so copies itself

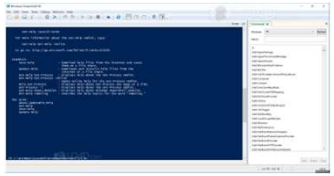
into the global
template file. This file is opened with every subsequent document, and the macro virus run, infecting that document. It then checks to see if it has been run on this system before, by looking to see if a specific key “Melissa” has been added to the registry. If that key is absent, and Outlook is the e-mail client, the macro virus then sends a copy of the current, infected document to each of the first 50 addresses in the current user’s Address Book. It then creates the “Melissa” registry entry, so this is only done once on any system. Finally it checks the current time and date for a specific trigger condition, which if met results in a Simpson quote being inserted into the current document.
Once the macro virus code has finished, the document continues opening and the user can then edit as normal. This code illustrates how a macro virus can manipulate both the document contents, and access other applications on the system. It also shows two infection mechanisms, the first infecting every subsequent document opened on the system, the second sending infected documents to other users via e-mail. More sophisticated macro virus code can use stealth techniques such as encryption or polymorphism, changing its appearance each time, to avoid scanning detection.

The lines are as follows.
Line 1. macro Document underscore Open
Line 2. indented once, disable Macro menu and some macro security features
Line 3. indented once, if called from a user document
Line 4. indented twice, copy macro code into Normal template file
Line 5. indented once, else
Line 6. indented twice, copy macro code into user document being opened
Line 7. indented once, end if
Line 8. indented once, if registry key double quotes Melissa double quotes not present
Line 9. indented twice, if Outlook is email client for
Line 10. indented thrice, first 50 addresses in address book
Line 11. indented four times, send email to that address
Line 12. Indented four times, with currently infected document attached
Line 13. indented thrice, end for
Line 14, indented twice, end if
Line 15. indented twice, create registry key double quotes Melissa double quotes
Line 16. indented once, end if
Line 17. indented once, if minute in hour equals day of month
Line 18. indented twice, insert text into document being opened

Line 19. indented once, end if
Line 20. end macro

Macros and scripts nowadays: PowerShell target

- These attacks can happen locally or remotely via a network connection, and they often leverage built-in tools like Windows PowerShell and Visual Basic, or Bash and Python on Linux systems. Even macros like those built into Microsoft's Office Suite can be leveraged by attackers.



Ms Powershell

PowerShell, the built-in Windows scripting language, is a popular target for malicious actors because of the powerful capabilities it provides. PowerShell allows remote and local execution, network access, and many other capabilities. In addition, since it is available by default on Windows systems and is often not carefully monitored, attackers can leverage it in many different ways, including for fileless malware attacks where PowerShell scripts are executed locally once a browser or plug-in is compromised.

Defenses against PowerShell attacks include using Constrained Language Mode, which limits sensitive commands in PowerShell, and using Windows Defender's built-in Application Control tool or AppLocker to validate scripts and to limit which modules and plug-ins can be run. It is also a good idea to turn on logging for PowerShell as well as Windows command-line auditing.

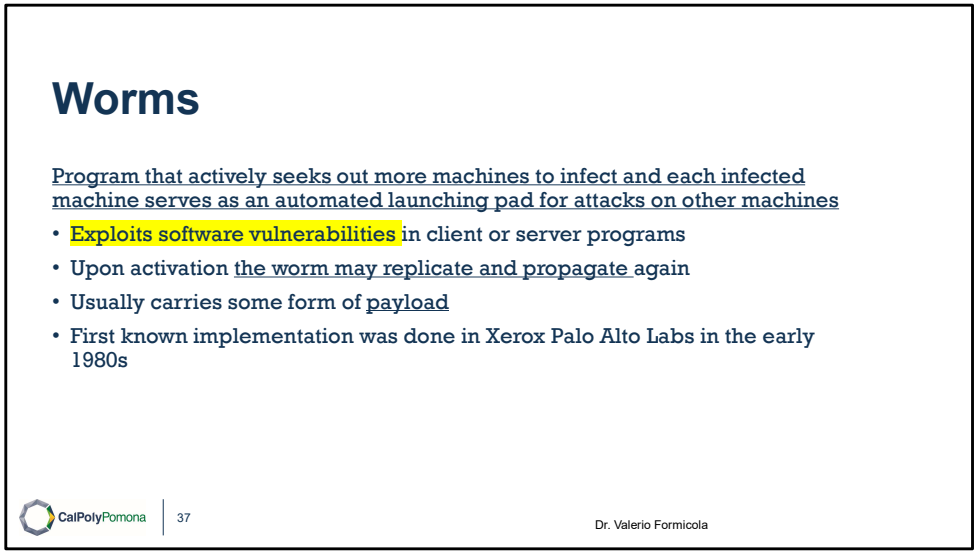
As a defender, enabling logging is one of the most important things you can do to make incident response easier. Make sure you consider whether you should have command-line and PowerShell logging turned on to allow you to detect attacks like those we discuss here.

Macros on Microsoft Office, Linux, MacOS

- Many Windows systems have Microsoft Office installed, and Microsoft Office macros written in **Visual Basic for Applications (VBA)** are another target for attackers. Although macro viruses are no longer as common as they once were, macros embedded in Office documents and similar functionality in other applications are potential targets for attackers, and if new vulnerabilities are discovered in Office, the popularity of macro viruses could increase.
- **Linux** systems are also targeted. Attackers may leverage common languages and tools like **Python** (it is ubiquitous in the Linux line of distributions and is available for installation on Windows machines), Perl, and **Bash** as part of their attack process. Languages like these can be used to create persistent remote access using bind or reverse shells, as well as a multitude of other useful exploit tools. **Metasploit**, a popular exploit tool, includes rootkits that leverage each of these languages.
- Preventing use of built-in or preexisting tools like programming languages and shells can be difficult because they are an important part of how users interact with and use the systems they exist on. That makes security that prevents attackers from gaining access to the systems through vulnerabilities, compromised accounts, and other means one of the most important layers of defense.

MacOS Python script: https://www.intego.com/mac-security-blog/beware-dangerous-macro-malware-ahead/
Example: writing Python malware https://hackmag.com/coding/python-malware/


Worms
CalPolyPomona

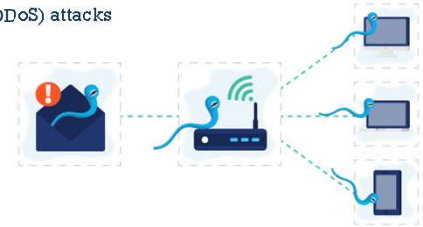


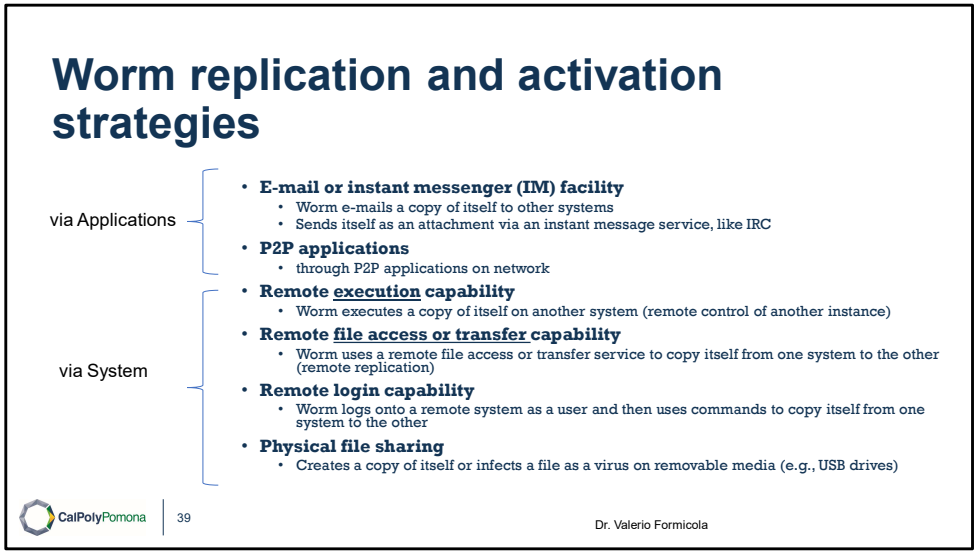
The next category of malware propagation concerns the exploit of software vulnerabilities, such as those we discuss in Chapters 10 and 11 , which are commonly exploited by computer worms. A worm is a program that actively seeks out more machines to infect, and then each infected machine serves as an automated launching pad for attacks on other machines. Worm programs exploit software vulnerabilities in client or server programs to gain access to each new system. They can use network connections to spread from system to system. They can also spread through shared media, such as USB drives or CD and DVD data disks. E-mail worms spread in macro or script code included in documents attached to e-mail or to instant messenger file transfers. Upon activation, the worm may replicate and propagate again. In addition to propagation, the worm usually carries some form of payload, such as those we discuss later.

The concept of a computer worm was introduced in John Brunner's 1975 SF novel *The Shockwave Rider . The first known worm implementation was done in* Xerox Palo Alto Labs in the early 1980s. It was nonmalicious, searching for idle systems to use to run a computationally intensive task.

Main characteristic of Worms: fast self-replicating and host independence

- A computer worm is a type of malware that can automatically propagate or self-replicate without human interaction, enabling its spread to other computers across a network.
- **Worms** spread from computer to computer and can move and operate independently. A worm's ability to send out hundreds or thousands of copies of itself is one of its biggest dangers.
- Typical uses of worms by hackers (payloads):
  - Launching distributed denial of service (DDoS) attacks
  - Conducting ransomware attacks
  - Stealing sensitive data
  - Dropping other malware
  - Consuming bandwidth
  - Deleting files
  - Overloading networks





To replicate itself, a worm uses some means to access remote systems. These include the following, most of which are still seen in active use:

• **Electronic mail or instant messenger facility:** A worm e-mails a copy of itself to other systems, or sends itself as an attachment via an of instant message service, so that its code is run when the e-mail or attachment is received or viewed.

• **File sharing:** A worm either creates a copy of itself or infects other suitable files as a virus on removable media such as a USB drive; it then executes when the drive is connected to another system using the autorun mechanism by exploiting some software vulnerability, or when a user opens the infected file on the target system.

• **Remote execution capability:** A worm executes a copy of itself on another system, either by using an explicit remote execution facility or by exploiting a program flaw in a network service to subvert its operations (as we discuss in Chapters 10 and 11).

• **Remote file access or transfer capability:** A worm uses a remote file access or transfer service to another system to copy itself from one system to the other, where users on that system may then execute it.

• **Remote login capability:** A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other, where it then executes.

The new copy of the worm program is then run on the remote system where, in addition to any payload functions that it performs on that system, it continues to propagate.

A worm typically uses the same phases as a computer virus: dormant, propagation, triggering, and execution. The propagation phase generally performs the following functions:

• Search for appropriate access mechanisms to other systems to infect by examining host tables, address books, buddy lists, trusted peers, and other similar repositories of remote system access details; by scanning possible target host addresses; or by searching for suitable removable media devices to use.

• Use the access mechanisms found to transfer a copy of itself to the remote system, and cause the copy to be run.

The worm may also attempt to determine whether a system has previously been infected before copying itself to the system. In a multiprogramming system, it can also disguise its presence by naming itself as a system process or using some other name that may not be noticed by a system operator. More recent worms can even inject their code into existing processes on the system, and run using additional threads in that process, to further disguise their presence.

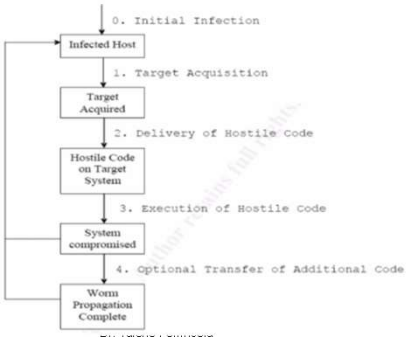
Danger of worms

- It's similar to a virus but way faster to replicate because doesn't need human intervention and is based on software vulnerabilities
- Due to fast replication, it's used in a broader range of attacks, including crashing systems through self-replication, downloading malicious applications, and providing hackers with backdoor access to equipment.
- Worms can also be hard to remediate. Because they spread automatically and quickly, it can take a lot of time and effort to eradicate a worm outbreak from the environment and fully recover. When a worm spreads inside a data storage environment, for example, it can take months to completely clean it up. Even when a worm doesn't have a malicious payload that does damage, it poses a serious nuisance for IT managers who have to dedicate valuable resources to navigate the incident response process.

Target Discovery (1 of 2)

First objective of a worm is to determine the victims. Hence, they use different strategies

- **Scanning (or fingerprinting)**
  - First function in the propagation phase for a network worm
  - Searches for other systems to infect
- **Random**
  - Each compromised host probes random addresses in the IP address space using a different seed
  - This produces a high volume of Internet traffic which may cause generalized disruption even before the actual attack is launched



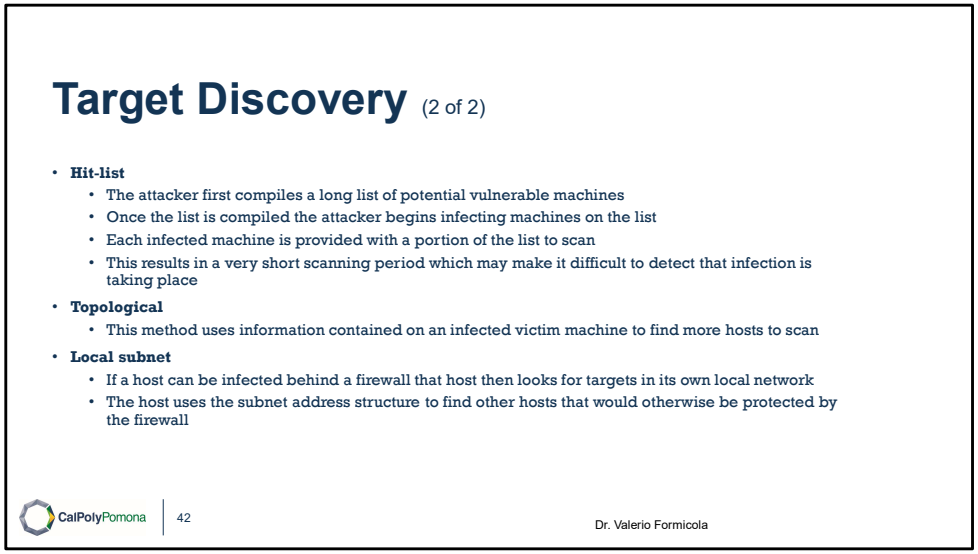
The first function in the propagation phase for a network worm is for it to search for other systems to infect, a process known as **scanning** or fingerprinting. For such worms, which exploit software vulnerabilities in remotely accessible network services, it must identify potential systems running the vulnerable service, and then infect them. Then, typically, the worm code now installed on the infected machines repeats the same scanning process, until a large distributed network of infected machines is created.

[MIRK04] lists the following types of network address scanning strategies that such a worm can use:

• **Random:** Each compromised host probes random addresses in the IP address space, using a different seed. This technique produces a high volume of Internet traffic, which may cause generalized disruption even before the actual attack is launched.

E-Mail Addresses.

- Host Lists.
- Trusted Systems.
- Network Neighborhood.
- DNS Queries.
- Randomly Selecting a Target Network Address

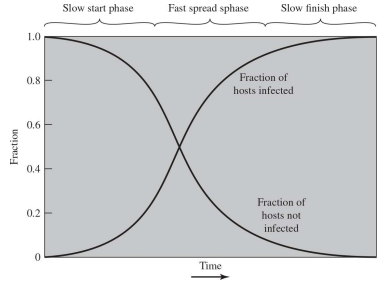


• **Hit-List:** The attacker first compiles a long list of potential vulnerable machines. This can be a slow process done over a long period to avoid detection that an attack is underway. Once the list is compiled, the attacker begins infecting machines on the list. Each infected machine is provided with a portion of the list to scan. This strategy results in a very short scanning period, which may make it difficult to detect that infection is taking place.

• **Topological:** This method uses information contained on an infected victim machine to find more hosts to scan.

• **Local subnet**: If a host can be infected behind a firewall, that host then looks for targets in its own local network. The host uses the subnet address structure to find other hosts that would otherwise be protected by the firewall.

Figure 6.2 Worm Propagation Model



$dI(t)dt=\beta I(t)\ S\ (t)$

where

I(t)=number of individuals infected as of time *t*

S(t)=number of susceptible individuals (susceptible to infection but not yet infected) at time *t*

β=infection rate

N=size of the population, N=I(t)+S(t)

A well-designed worm can spread rapidly and infect massive numbers of hosts. It is useful to have a general model for the rate of worm propagation. Computer viruses and worms exhibit similar self-replication and propagation behavior to biological viruses. Thus we can look to classic epidemic models for understanding computer virus and worm propagation behavior.

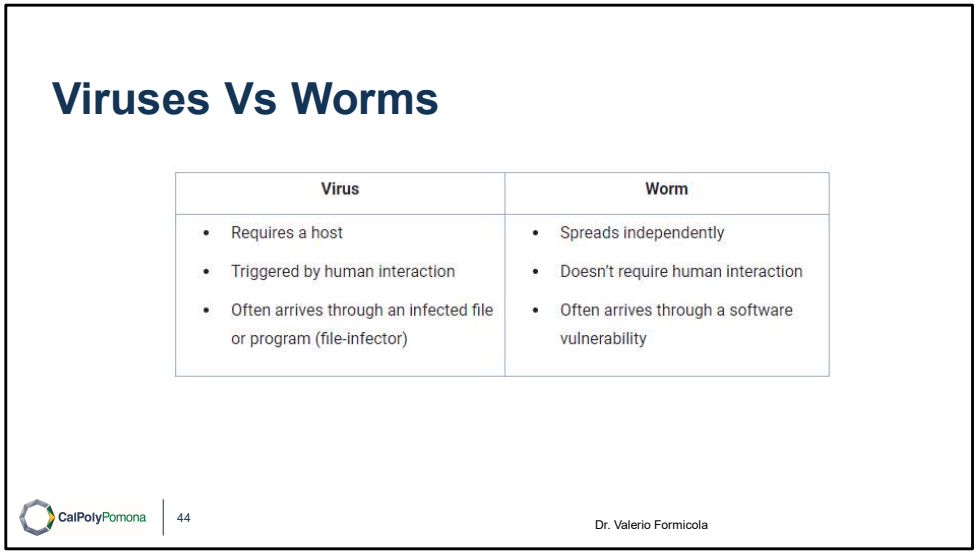
Figure 6.2 shows the dynamics of worm propagation using this model. Propagation proceeds through three phases. In the initial phase, the number of hosts increases exponentially. To see that this is so, consider a simplified case in which a worm is launched from a single host and infects two nearby hosts. Each of these hosts infects two more hosts, and so on. This results in exponential growth. After a time, infecting hosts waste some time attacking already infected hosts, which reduces the rate of infection. During this middle phase, growth is approximately linear, but the rate of infection is rapid. When most vulnerable computers have been infected, the attack enters a slow finish phase as the worm seeks out those remaining hosts that are difficult to identify.

Clearly, the objective in countering a worm is to catch the worm in its slow start phase, at a time when few

hosts have been infected.

Zou, et al [ZOU05] describe a model for worm propagation based on an analysis of network worm attacks at that time. The speed of propagation and the total number of hosts infected depend on a number of factors, including the mode of propagation, the vulnerability or vulnerabilities exploited, and the degree of similarity to preceding attacks. For the latter factor, an attack that is a variation of a recent previous attack may be countered more effectively than a more novel attack. Zou's model agrees closely with Figure 6.2.

The vertical axis ranges from 0 to 1.0, in increments of 0.2. The horizontal axis is labeled, slow start phase, fast spread phase, and slow finish phase. The graph for fraction of hosts infected rises from (slow start phase, 0), passes through (fast spread phase, 0.5), and ends at (slow finish phase, 1.0). The graph for fraction of hosts not infected falls from (slow start phase, 1.0), passes through (fast finish phase, 0.5), and ends at (slow finish phase, 0).



Here's a typical example of getting a worm infection: You get a notification that Windows has a critical security update. But you're busy doing something else, so you ignore it, and then forget to install it later. That update was intended to fix a security vulnerability, but since you didn't apply the update, the hole (vulnerability) remains in your system.

Sooner or later, an enterprising hacker finds this hole and exploits it. Then, while you're busy working or gaming, a worm burrows into your computer and begins replicating itself, compromising your data and causing all kinds of other damage.

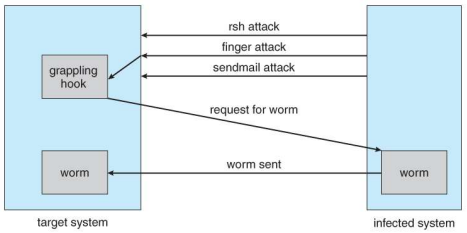
Before you have a chance to realize what's happening, the worm scans your network. It finds that you haven't applied the update on your other computer either, and neither has your spouse. The worm quickly spreads to all those devices, too.

Still unaware of any problem, you decide to head out for a cup of coffee. You sit down with your laptop at your local café and connect to their Wi-Fi. The worm scans the coffee shop's network, finding and infecting a dozen more devices (and people) that have the same vulnerability. Those people eventually go home, and their devices then infect their family members' devices too, and so on.

Example: Morris Worm

- Earliest significant worm infection
- Released by Robert Morris in 1988
- Designed to spread on UNIX systems.
- Part 1: A small program called a grappling hook is deposited on the target using 3 vulnerabilities:
  - Attempted to crack local password file to use login/password to logon to other systems
  - Exploited a bug in the finger protocol which reports the whereabouts of a remote user
  - Exploited a trapdoor in the debug option of the remote process that receives and sends mail
- Part 2: Successful attacks achieved communication with the operating system command interpreter
  - Sent interpreter a bootstrap program to copy worm over

- Born on November 8, 1965
- A Harvard Graduate and attended graduate school at Cornell
- First person convicted under the new "Computer Fraud and Abuse Act".



Arguably, the earliest significant, and hence well-known, worm infection was released onto the Internet by Robert Morris in 1988 [ORMA03]. The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation. When a copy began execution, its first task was to discover other hosts known to this host that would allow entry from this host. The worm performed this task by examining a variety of lists and tables, including system tables that declared which other machines were trusted by this host, users' mail forwarding files, tables by which users gave themselves permission for access to remote accounts, and from a program that reported the status of network connections. For each discovered host, the worm tried a number of methods for gaining access:

1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password-cracking program that tried

a. Each user's account name and simple permutations of it

b. A list of 432 built-in passwords that Morris thought to be likely candidates
c. All the words in the local system dictionary

2. It exploited a bug in the UNIX finger protocol, which reports the whereabouts of a remote user.

3. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter. It then sent this interpreter a short bootstrap program, issued a command to execute that program, and then logged off. The bootstrap program then called back the parent program and downloaded the remainder of the worm. The new worm was then executed.

Another explanation:

•The three vulnerabilities exploited by the Morris Internet worm were as follows:

- **rsh (remote shell)** is a utility that was in common use at that time for accessing remote systems without having to provide a password. If a user had an account on two different computers (with the same account name on both systems), then the system could be configured to allow that user to remotely connect from one system to the other without having to provide a password. Many systems were configured so that ***any*** user (except root) on system A could access the same account on system B without providing a password.
- **finger** is a utility that allows one to remotely query a user database, to find the true name and other information for a given account name on a given system. For example "finger joeUser@somemachine.edu" would access the finger daemon at somemachine.edu and return information regarding joeUser. Unfortunately the finger daemon (which ran with system privileges) had the buffer overflow problem, so by sending a special 536-character user name the worm was able to fork a shell on the remote system running with root privileges.
- **sendmail** is a routine for sending and forwarding mail that also included a debugging option for verifying and testing the system. The debug feature was convenient for administrators, and was often left turned on. The Morris worm exploited the debugger to mail and execute a copy of the grappling hook program on the remote system.

•Once in place, the worm undertook systematic attacks to discover user passwords:

- First it would check for accounts for which the account name and the password were the same, such as "guest", "guest".
- Then it would try an internal dictionary of 432 favorite password choices. (I'm sure "password", "pass", and blank

passwords were all on the list.)

- Finally it would try every word in the standard UNIX on-line dictionary to try and break into user accounts.

•Once it had gotten access to one or more user accounts, then it would attempt to use those accounts to rsh to other systems, and continue the process.

•With each new access the worm would check for already running copies of itself, and 6 out of 7 times if it found one it would stop. (The seventh was to prevent the worm from being stopped by fake copies.)

•Fortunately the same rapid network connectivity that allowed the worm to propagate so quickly also quickly led to its demise - Within 24 hours remedies for stopping the worm propagated through the Internet from administrator to administrator, and the worm was quickly shut down.

•There is some debate about whether Mr. Morris's actions were a harmless prank or research project that got out of hand or a deliberate and malicious attack on the Internet. However the court system convicted him, and penalized him heavy fines and court costs.

•There have since been many other worm attacks, including the W32.Sobig.F@mm attack which infected hundreds of thousands of computers and an estimated 1 in 17 e-mails in August 2003. This worm made detection difficult by varying the subject line of the infection-carrying mail message, including "Thank You!", "Your details", and "Re: Approved".

Recent Worm Attacks

Considering Melissa, the propagation part is the same of worms

Melissa	1998	E-mail worm First to include virus, worm and Trojan in one package
Code Red	July 2001	Exploited Microsoft IIS bug (Microsoft Internet Information Server, a web server) Probes random IP addresses Consumes significant Internet capacity when active
Code Red II	August 2001	Also targeted Microsoft IIS Installs a backdoor for access
Nimda	September 2001	Had worm, virus and mobile code characteristics Spread using e-mail, Windows shares, Web servers, Web clients, backdoors
SQL Slammer	Early 2003	Exploited a buffer overflow vulnerability in SQL server compact and spread rapidly
Sobig.F	Late 2003	Exploited open proxy servers to turn infected machines into spam engines
Mydoom	2004	Mass-mailing e-mail worm Installed a backdoor in infected machines
Warezov	2006	Creates executables in system directories Sends itself as an e-mail attachment Can disable security related products
Conficker (Downadup)	November 2008	Exploits a Windows buffer overflow vulnerability Most widespread infection since S QL Slammer
Stuxnet	2010	Restricted rate of spread to reduce chance of detection Targeted industrial control systems

CalPolyPomona | 46 | Dr. Valerio Formicola

The Melissa e-mail worm that appeared in 1998 was the first of a new generation of malware that included aspects of virus, worm, and Trojan in one package [CASS01]. Melissa made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then it
1. Sends itself to everyone on the mailing list in the user's e-mail package, propagating as a worm; and

2. Does local damage on the user's system, including disabling some security tools, and also copying itself into other documents, propagating as a virus; and

3. If a trigger time was seen, it displayed a Simpson quote as its payload.

In 1999, a more powerful version of this e-mail virus appeared. This version could be activated merely by opening an e-mail that contains the virus, rather than by opening an attachment. The virus uses the Visual Basic scripting language supported by the e-mail package.

Melissa propagates itself as soon as it is activated (either by opening an e-mail attachment or by opening the e-

mail) to all of the e-mail addresses known to the infected host. As a result, whereas viruses used to take months or years to propagate, this next generation of malware could do so in hours. [CASS01] notes that it took only three days for Melissa to infect over 100,000 computers, compared to the months it took the Brain virus to infect a few thousand computers a decade before. This makes it very difficult for anti-virus software to respond to new attacks before much damage is done.

The Code Red worm first appeared in July 2001. Code Red exploits a security hole in the Microsoft Internet Information Server (IIS) to penetrate and spread. It also disables the system file checker in Windows. The worm probes random IP addresses to spread to other hosts. During a certain period of time, it only spreads. It then initiates a denial-of-service attack against a government Web site by flooding the site with packets from numerous hosts. The worm then suspends activities and reactivates periodically. In the second wave of attack, Code Red infected nearly 360,000 servers in 14 hours. In addition to the havoc it caused at the targeted server, Code Red consumed enormous amounts of Internet capacity, disrupting service [MOOR02].

Code Red II is another, distinct, variant that first appeared in August 2001, and also targeted Microsoft IIS. It tried to infect systems on the same subnet as the infected system. Also, this newer worm installs a backdoor, allowing a hacker to remotely execute commands on victim computers.

The Nimda worm that appeared in September 2001 also has worm, virus, and mobile code characteristics. It spread using a variety of distribution methods:

• **E-mail**: A user on a vulnerable host opens an infected e-mail attachment; Nimda looks for e-mail addresses on the host and then sends copies of itself to those addresses.

• **Windows shares**: Nimda scans hosts for unsecured Windows file shares; it can then use NetBIOS86 as a transport mechanism to infect files on that host in the hopes that a user will run an infected file, which will activate Nimda on that host.

• **Web servers**: Nimda scans Web servers, looking for known vulnerabilities in Microsoft IIS. If it finds a vulnerable server, it attempts to transfer a copy of itself to the server and infects it and its files.

Web clients: If a vulnerable Web client visits a Web server that has been infected by Nimda, the client's workstation

will become infected.

• **Backdoors**: If a workstation was infected by earlier worms, such as “Code Red II,” then Nimda will use the backdoor access left by these earlier infections to access the system.

In early 2003, the SQL Slammer worm appeared. This worm exploited a buffer overflow vulnerability in Microsoft SQL server. The Slammer was extremely compact and spread rapidly, infecting 90% of vulnerable hosts within 10 minutes. This rapid spread caused significant congestion on the Internet.

Late 2003 saw the arrival of the Sobig.F worm, which exploited open proxy servers to turn infected machines into spam engines. At its peak, Sobig.F reportedly accounted for one in every 17 messages and produced more than one million copies of itself within the first 24 hours.

Mydoom is a mass-mailing e-mail worm that appeared in 2004. It followed a growing trend of installing a backdoor in infected computers, thereby enabling hackers to gain remote access to data such as passwords and credit card numbers. Mydoom replicated up to 1,000 times per minute and reportedly flooded the Internet with 100 million infected messages in 36 hours.

The Warezov family of worms appeared in 2006 [KIRK06]. When the worm is launched, it creates several executables in system directories and sets itself to run every time Windows starts by creating a registry entry. Warezov scans several types of files for e-mail addresses and sends itself as an e-mail attachment. Some variants are capable of downloading other malware, such as Trojan horses and adware. Many variants disable security-related products and/or disable their updating capability.

The Conficker (or Downadup) worm was first detected in November 2008 and spread quickly to become one of the most widespread infections since SQL Slammer in 2003 [LAWT09]. It spread initially by exploiting a Windows buffer overflow vulnerability, though later versions could also spread via USB drives and network file shares. In 2010, it still comprised the second most common family of malware observed by Symantec [SYMA16], even though patches were available from Microsoft to close the main vulnerabilities it exploits.

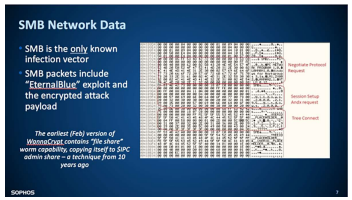
In 2010, the Stuxnet worm was detected, though it had been spreading quietly for some time previously [CHEN11, KUSH13]. Unlike many previous worms, it deliberately restricted its rate of spread to reduce its chance of detection. It

also targeted industrial control systems, most likely those associated with the Iranian nuclear program, with the likely aim of disrupting the operation of their equipment. It supported a range of propagation mechanisms, including via USB drives, network file shares, and using no less than four unknown, zero-day vulnerability exploits. Considerable debate resulted from the size and complexity of its code, the use of an unprecedented four zero-day exploits, and the cost and effort apparent in its development. There are claims that it appears to be the first serious use of a cyberwarfare weapon against a nation's physical infrastructure. The researchers at Symantec who analyzed Stuxnet noted that while they were expecting to find espionage, they never expected to see malware with targeted sabotage as its aim. As a result, greater attention is now being directed at the use of malware as a weapon by a number of nations.

In late 2011 the Duqu worm was discovered, which uses code related to that in Stuxnet. Its aim is different, being cyber-espionage, though it appears to also target the Iranian nuclear program. Another prominent, recent, cyber-espionage worm is the Flame family, which was discovered in 2012 and appears to target Middle-Eastern countries. Despite the specific target areas for these various worms, their infection strategies have been so successful that they have been identified on computer systems in a very large number of countries, including on systems kept physically isolated from the general Internet. This reinforces the need for significantly improved countermeasures to resist such infections.

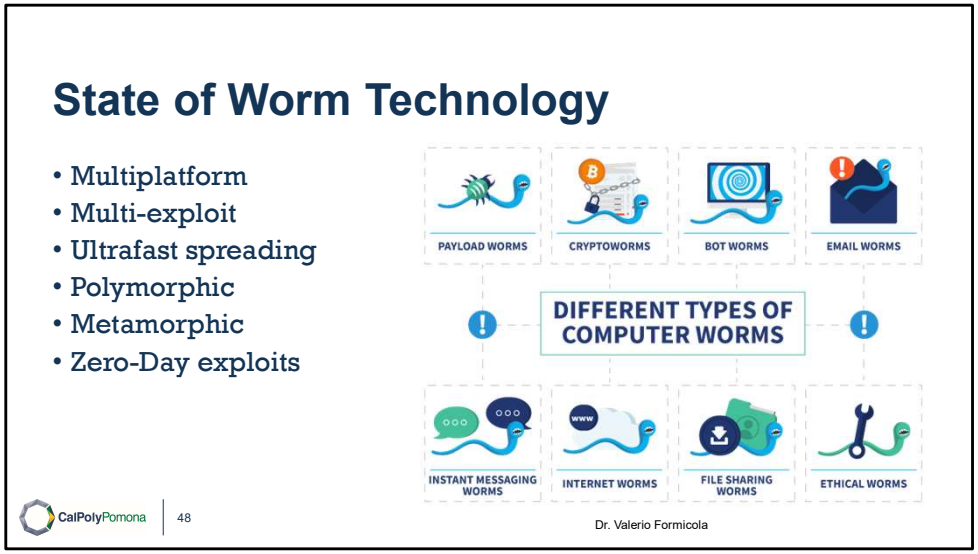
Example: WannaCry

- Ransomware attack in May 2017 that spread extremely fast over a period of hours to days, infecting hundreds of thousands of systems belonging to both public and private organizations in more than 150 countries
- It spread as a worm by aggressively scanning both local and random remote networks, attempting to exploit a vulnerability in the SMB file sharing service on unpatched Windows systems
- This rapid spread was only slowed by the accidental activation of a "kill-switch" domain by a UK security researcher
- Once installed on infected systems, it also encrypted files, demanding a ransom payment to recover them



Dr. Valerio Formicola

In May 2017, the WannaCry ransomware attack spread extremely rapidly over a period of hours to days, infecting hundreds of thousands of systems belonging to both public and private organisations in more than 150 countries (US-CERT Alert TA17- 132A) [GOOD17]. It spread as a worm by aggressively scanning both local and random remote networks, attempting to exploit a vulnerability in the SMB file sharing service on unpatched Windows systems. This rapid spread was only slowed by the accidental activation of a "kill-switch" domain by a UK security researcher, whose existence was checked for in the initial versions of this malware. Once installed on infected systems, it also encrypted files, demanding a ransom payment to recover them, as we will discuss later.



The state of the art in worm technology includes the following:

• **Multiplatform**: Newer worms are not limited to Windows machines but can attack a variety of platforms, especially the popular varieties of UNIX; or exploit macro or scripting languages supported in popular document types.

• **Multi-exploit**: New worms penetrate systems in a variety of ways, using exploits against Web servers, browsers, e-mail, file sharing, and other network-based applications; or via shared media.

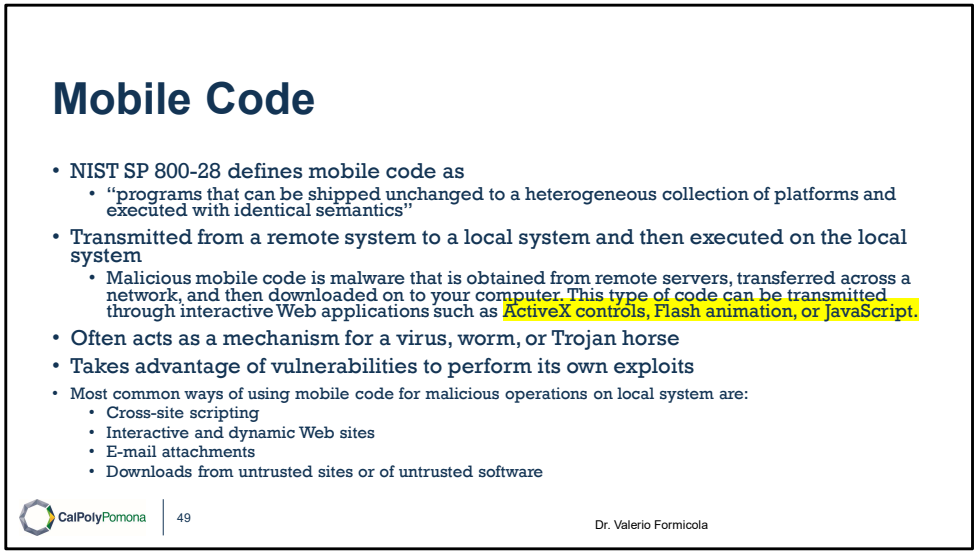
• **Ultrafast spreading**: Exploit various techniques to optimize the rate of spread of a worm to maximize its likelihood of locating as many vulnerable machines as possible in a short time period.

• **Polymorphic**: To evade detection, skip past filters, and foil real-time analysis, worms adopt the virus polymorphic technique. Each copy of the worm has new code generated on the fly using functionally equivalent instructions and encryption techniques.

• **Metamorphic**: In addition to changing their appearance, metamorphic worms have a repertoire of behavior patterns that are unleashed at different stages of propagation.

• **Transport vehicles**: Because worms can rapidly compromise a large number of systems, they are ideal for spreading a wide variety of malicious payloads, such as distributed denial-of-service bots, rootkits, spam e-mail generators, and spyware.

• **Zero-day exploi**t : To achieve maximum surprise and distribution, a worm should exploit an unknown vulnerability that is only discovered by the general network community when the worm is launched. In 2015, 54 zero-day exploits were discovered and exploited, significantly more than in previous years [SYMA16]. Many of these were in common computer and mobile software. Some, though, were in common libraries and development packages, and some in industrial control systems. This indicates the range of systems being targeted.



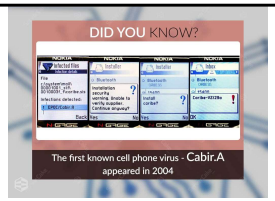
NIST SP 800-28 (Guidelines on Active Content and Mobile Code , March 2008) defines mobile code as programs (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and executed with identical semantics.

Mobile code is transmitted from a remote system to a local system and then executed on the local system without the user's explicit instruction [SOUP13]. Mobile code often acts as a mechanism for a virus, worm, or Trojan horse to be transmitted to the user's workstation. In other cases, mobile code takes advantage of vulnerabilities to perform its own exploits, such as unauthorized data access or root compromise. Popular vehicles for mobile code include Java applets, ActiveX, JavaScript, and VBScript. The most common ways of using mobile code for malicious operations on local system are cross-site scripting, interactive and dynamic Web sites, e-mail attachments, and downloads from untrusted sites or of untrusted software.

malicious mobile code criminals program codes that install malware into items of interest such as free screensavers, music downloads, games, pornography, and other applications that are accessed on the Internet. All of these applications generally require interactive plug-ins such as ActiveX, JavaScript, or Flash, and they exist on websites that are infected with malware.

Once the user clicks on the website and uses these applications, the malware is installed without the user's permission and is usually the initial step to a combined malware attack. The malware is installed on the user's computer and then it generates additional malware such as spyware, keylogging, adware, and other malicious software. This allows the intruder to access personal and financial information, passwords, logins, and other sensitive data.

Mobile Phone Worms



- First discovery was Cabir worm in 2004
- Then Lasco and CommWarrior in 2005
- Communicate through **Bluetooth** wireless connections or **MMS**
- **Target is the smartphone**
- Can completely disable the phone, delete data on the phone, or force the device to send costly messages
- CommWarrior replicates by means of Bluetooth to other phones, sends itself as an MMS file to contacts and as an auto reply to incoming text messages

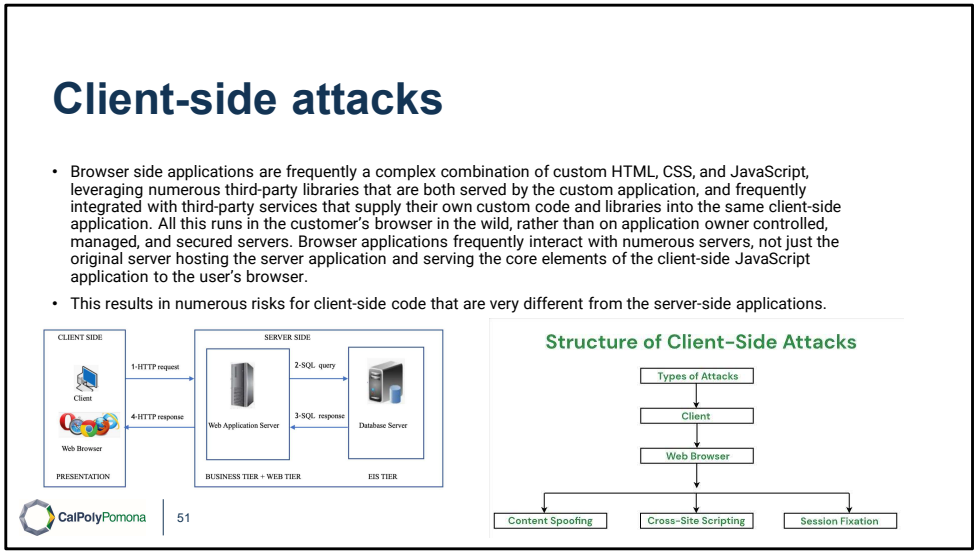
CalPolyPomona | 50 Dr. Valerio Formicola

Worms first appeared on mobile phones with the discovery of the Cabir worm in 2004, and then Lasco and CommWarrior in 2005. These worms communicate through Bluetooth wireless connections or via the multimedia messaging service (MMS). The target is the smartphone, which is a mobile phone that permits users to install software applications from sources other than the cellular network operator. All these early mobile worms targeted mobile phones using the Symbian operating system. More recent malware targets Android and iPhone systems. Mobile phone malware can completely disable the phone, delete data on the phone, or force the device to send costly messages to premium-priced numbers.

The CommWarrior worm replicates by means of Bluetooth to other phones in the receiving area. It also sends itself as an MMS file to numbers in the phone's address book and in automatic replies to incoming text messages and MMS messages. In addition, it copies itself to the removable memory card and inserts itself into the program installation files on the phone.

Although these examples demonstrate that mobile phone worms are possible, the vast

majority of mobile phone malware observed use trojan apps to install themselves [SYMA16].



https://owasp.org/www-project-top-10-client-side-security-risks/

A client-side attack is a security breach that happens on the client side. Examples include installing malware on your device or banking credentials being stolen by third-party sites. A common client-side attack is a denial of service attack, which floods a system with requests and prevents it from functioning properly. For example, if you tried to log into your bank's website using an out-of-date browser and plugin, you would be denied access. Another common client-side attack is data manipulation, for example, changing your bank balance without your permission. On the other hand, a security breach on the client side should be prevented by using positive client-side security measures such as multifactor authentication and encryption. Ideally, computers should only be connected to trusted networks and devices with proper security patches installed. Client-side attacks are harder to prevent since they require access to your device, but can be mitigated by taking appropriate steps beforehand.

Types of Client-Side Attacks:

•**Content Spoofing**: Content Spoofing is one of the common web security vulnerabilities. It allows the end

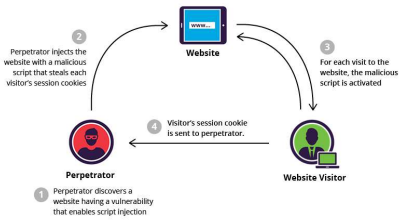
user of the vulnerable web application to spoof or modify the actual content on the web page. The user might use the security loopholes on the website to inject the content that he/she wishes into the target website. When an application does not properly handle user-supplied data, an attacker can supply content to a web application, typically via a parameter value, that is reflected back to the user.

•**Cross-site scripting**: In this type of attack, an attacker injects malicious codes into websites that are typically downloaded and displayed by a vulnerable browser. Cross-site scripting (XSS) is a computer security vulnerability typically found in web applications that enable attackers to inject client-side script into web pages viewed by other users.

•**Session Fixation Attack**: A session fixation attack is a type of remote code execution attack which is used to exploit software designed with the web-server Session Management feature. When a website is running an HTTP server, the server's session state information can be stolen and then retrieved by an attacker to take over the browser or use it for further attacks. There are many tools that can help you detect session fixation attacks in your organization in order to prevent future attacks. A Session fixation attack is also known as Session Fixation Vulnerability (SFV).

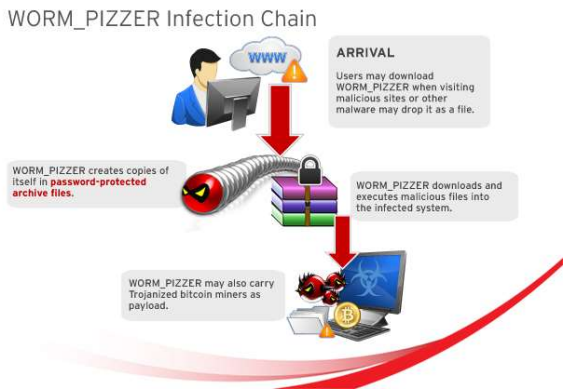
Drive-By-Downloads

- Exploits browser and plugin vulnerabilities so when the user views a webpage controlled by the attacker, it contains code that exploits the bug to download and install malware on the system without the user's knowledge or consent
- In most cases the malware does not actively propagate as a worm does
- Spreads when users visit the malicious Web page
- Exploits vulnerabilities of clients while using Adobe Flash Player and Oracle Java plugins



Another approach to exploiting software vulnerabilities involves the exploit of bugs in user applications to install malware. A common technique exploits browser vulnerabilities so that when the user views a Web page controlled by the attacker, it contains code that exploits the browser bug to download and install malware on the system without the user's knowledge or consent. This is known as a **drive-by-download** and is a common exploit in recent attack kits. Multiple vulnerabilities in the Adobe Flash Player and Oracle Java plugins have been exploited by attackers over many years, to the point where many browsers are now removing support for them. In most cases, this malware does not actively propagate as a worm does, but rather waits for unsuspecting users to visit the malicious webpage in order to spread to their systems [SYMA16].

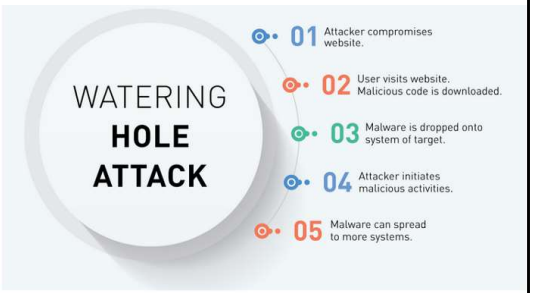
Example



https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/121/worm-spreads-across-passwordprotected-archive-files

Watering-Hole Attacks

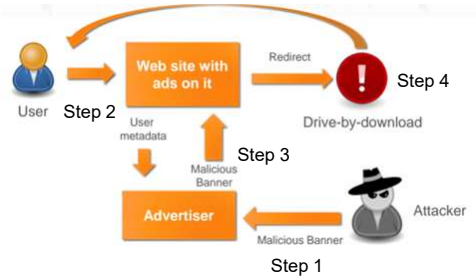
- A variant of drive-by-download used in highly targeted attacks
- The attacker researches their intended victims to identify websites they are likely to visit, then scans these sites to identify those with vulnerabilities that allow their compromise
- They then wait for one of their intended victims to visit one of the compromised sites
- Attack code may even be written so that it will only infect systems belonging to **the target organization** and take no action for other visitors to the site
- This greatly increases the likelihood of the site compromise remaining undetected



In general, drive-by-download attacks are aimed at anyone who visits a compromised site and is vulnerable to the exploits used. **Watering-hole attacks** are a variant of this used in highly targeted attacks. The attacker researches their intended victims to identify web sites they are likely to visit, and then scans these sites to identify those with vulnerabilities that allow their compromise with a drive-by-download attack. They then wait for one of their intended victims to visit one of the compromised sites. Their attack code may even be written so that it will only infect systems belonging to the **target organization**, and take no action for other visitors to the site. This greatly increases the likelihood of the site compromise remaining undetected.

Malvertising

- Places **malware on websites without actually compromising them**
- The attacker pays for advertisements that are highly likely to be placed on their intended target websites and incorporate malware in them
- Using these malicious ads, attackers can infect visitors to sites displaying them
- The malware code may be dynamically generated to either reduce the chance of detection or to only infect specific systems
- Has grown rapidly in recent years because they are easy to place on desired websites with few questions asked and are hard to track
- Attackers can place these ads for as little as a few hours, when they expect their intended victims could be browsing the targeted websites, greatly reducing their visibility

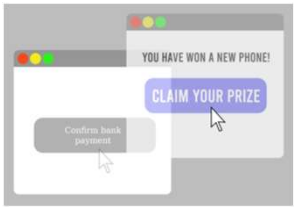


Malvertising is another technique used to place malware on websites without actually compromising them. The attacker pays for advertisements that are highly likely to be placed on their intended target websites, and which incorporate malware in them. Using these malicious adds, attackers can infect visitors to sites displaying them. Again, the malware code may be dynamically generated to either reduce the chance of detection, or to only infect specific systems. Malvertising has grown rapidly in recent years, as they are easy to place on desired websites with few questions asked, and are hard to track. Attackers have placed these ads for as little as a few hours, when they expect their intended victims could be browsing the targeted websites, greatly reducing their visibility [SYMA16].

Other malware may target common PDF viewers to also download and install malware without the user's consent when they view a malicious PDF document [STEV11]. Such documents may be spread by spam e-mail, or be part of a targeted phishing attack, as we will discuss in the next section.

Clickjacking

- Also known as a ***user-interface (UI) redress attack***
- Using a similar technique, keystrokes can also be hijacked
  - A user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker



Dr. Valerio Formicola

Clickjacking, also known as a *user-interface (UI) redress attack* , is a vulnerability used by an attacker to collect an infected user's clicks. The attacker can force the user to do a variety of things from adjusting the user's computer settings to unwittingly sending the user to Web sites that might have malicious code. Also, by taking advantage of Adobe Flash or JavaScript, an attacker could even place a button under or over a legitimate button, making it difficult for users to detect. A typical attack uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is hijacking clicks meant for one page and routing them to another page, most likely owned by another application, domain, or both.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

There is a wide variety of techniques for accomplishing a clickjacking attack, and new techniques are developed as defenses to older techniques are put in place. [NIEM11] and [STON10] are useful discussions.

Vulnerability used by an attacker to collect an infected user's clicks

- The attacker can force the user to do a variety of things from adjusting the user's computer settings to unwittingly sending the user to Web sites that might have malicious code
- By taking advantage of Adobe Flash or JavaScript an attacker could even place a button under or over a legitimate button making it difficult for users to detect
- A typical attack uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page
- The attacker is hijacking clicks meant for one page and routing them to another page