FIT5037: Network Security
Computer system security and malicious code

Faculty of Information Technology
Monash University

May 7, 2019

Copyright Notice

Lecture 9: Computer system security and malicious code

Lecture Topics:

- Symmetric key cryptography
- Asymmetric key cryptography
- Pseudorandom Number Generators and hash functions
- Authentication Methods and AAA protocols
- Security at Network layer (IPsec)
- Security at Network layer (firewalls and wireless security)
- Security at Transport layer
- Security at Application layer
- **Computer system security and malicious code**
- Computer system vulnerabilities and penetration testing
- Intrusion detection
- Denial of Service Attacks and Countermeasures / Revision

Outline

- Malicious code definition and types
- Remote code execution
- Malware incident prevention
  - policy
  - awareness
  - Vulnerability mitigation
  - Threat mitigation
  - Defensive architecture
- Malware incident response
  - Preparation
  - Detection and analysis
  - Eradication and recovery
  - Post-incident activities

Malicious Code: Definition

- RFC 4949: Internet Security Glossary, Version 2 defines "malicious logic" as:

Hardware, firmware, or software that is intentionally included or inserted in a system for a harmful purpose.

- the RFC categorises the potential impact of malicious logic as:
  - corruption
  - incapacitation
  - masquerade
  - and misuse
- examples of malicious logic in RFC4949:
  - virus
  - worm
  - logic bomb
  - Trojan horse
  - spyware (deprecated term due to broad classification)

Malicious Code: Types

- virus
  - a self-replicating (and usually hidden) section of computer software (usually malicious logic)
  - propagates by infecting i.e., inserting a copy of itself into and becoming part of another program
    - A virus cannot run by itself; it requires that its host program be run to make the virus active.
  - NIST SP 800-83r1 divides viruses into two classes
    - compiled: executed by OS such as file infector viruses and boot sector viruses
    - interpreted: executed by an application such as macro viruses infecting application documents
- worm
  - a computer program that can run independently
  - propagates a complete working version of itself onto other hosts on a network
    - may consume system resources destructively
  - NIST SP 800-83r1 divides worms into two categories
    - Network service worms which exploits a network service vulnerability to propagate itself
    - Mass mailing worms which is a self contained malicious code that uses email to propagate itself
- logic bomb
  - activates when specified conditions are met
  - Usually intended to cause denial of service or otherwise damage system resources.

Malicious Code: Types (continued)

- Trojan horse
  - a computer program that appears to have a useful function
  - also has a hidden and potentially malicious function that evades security mechanisms
    - sometimes by exploiting legitimate authorizations of a system entity that invokes the program.
- spyware
  - this term is widely used to describe different malicious logic (deprecated according to RFC4949 for this reason)
  - Software that an intruder has installed surreptitiously (secretly) on a networked computer to:
    - gather data from that computer
    - and send it through the network to the intruder or some other interested party.
- ransomware
  - a type of malicious code that blocks or limits access to a system until a ransom is paid
    - lockers: lock down the system
    - encryptors: encrypt specific files e.g. WannaCry
- malicious mobile code (NIST)
  - software with malicious intent that is transmitted from a remote host to a local host (hence mobile)
  - executed locally without the user's explicit instruction
    - popular languages: Java, ActiveX, JavaScript, and VBScript
- blended attacks (NIST)
  - use of multiple infection or transmission methods

Attacker tools

NIST SP 800-83r1 provides a list of attacker tools that may be delivered to an infected host

- backdoors
  - malicious code that listens for commands on a certain TCP or UDP port
  - allows an attacker to perform certain tasks e.g. acquire passwords or execute commands
  - Zombies or Botnets are a collection of hosts with backdoors under an attacker control
- key loggers
  - monitors and records keyboard use
  - attacker may retrieve data or key logger may transfer data to the command and control host
- rootkits
  - collection of files to change the functionality of the infected host in a malicious way
  - tries to hide its existence
- web browser plug-ins
  - malicious plug-ins can monitor all use of a browser
- email generators
  - can be used as a propagation engine for the attacker to infect other hosts in infected host network
- attacker tool kit
  - different types of utilities such as packet sniffers, port scanners, vulnerability scanners, password crackers etc.
  - Offensive security tools: e.g. Wireshark, Kali Linux, nmap, openvas, Nessus, John the Ripper etc.

Malicious code obfuscation

One of the main approaches of anti-virus applications is signature detection, to evade this security measure malicious code use obfuscation techniques

- encryption: the body of the malicious code is encrypted except for a small part that performs decryption
  - the encryption key changes with each infection hence signature (bit pattern) of the code changes
  - the key must be stored with the virus

Polymorphic mutation

- encryption + decryptor mutation
  - in addition to encrypting its code it mutates its decryptor using other obfuscation techniques

Metamorphic mutation techniques (changing appearance/signature/bit pattern)

- insertion of no operation or useless instructions (junk code)
  - malicious code has a mutation engine that changes the signature of the code with each infection
  - the code changes by insertion of no operation instructions between malicious code instructions
    - most instructions can be used here as longs the inserted instruction does not change the logic

Malicious code obfuscation (continued)

Metamorphic mutation techniques ...

- register renaming/variable substitution
  - most processors provide multiple registers to store variables and intermediate values
    - mutation engine changes the registers used in each instruction without changing the logic
- subroutine permutation
  - the order of defined subroutines are changed which does not affect the logic of the code
  - the overall bit pattern of the code changes
- code reordering through jumps
  - the code is divided into sections and permuted and jump instructions are inserted to preserve logic
- equivalent code substitution/instruction replacement
  - part of the code (one or more instructions) is replaced by an equivalent code with different set of instructions (different bit pattern)

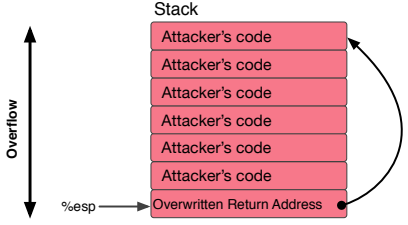
Remote Code Execution

- attacker exploits a vulnerability and is able to execute code on the target system
- examples of vulnerabilities that could lead to remote code execution
  - stack-based buffer overflow
  - heap-based buffer overflow
  - command injection
- exploitation techniques
  - shell code injection
  - return-to-libc (and return to library in general)
  - Return Oriented Programming
  - Jump Oriented Programming
- exploitation methods can be built into the malicious code as infection and propagation method

Remote Code Execution: stack-based BOF shell code injection

- when a function is called current function state is pushed unto the stack
- problem is some control information is also pushed to the stack
  - current function stack frame address
  - return address (to continue execution after the call)
- buffer over flow vulnerability
  - allows the user to write beyond assigned memory cells to a variable

- shell code injection
  - attacker injects instructions into the stack (overflow buffer)
    - generally to call system to execute a shell
  - attacker overwrites the return address with the address of the injected code



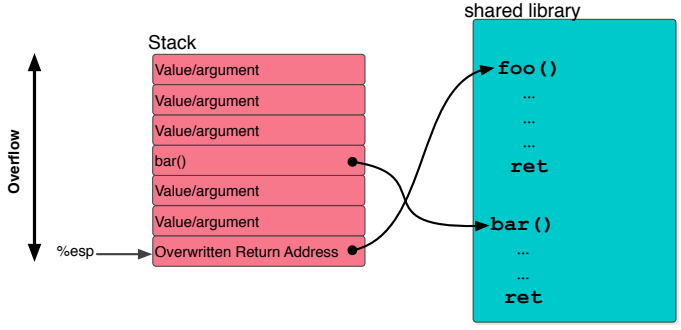
Remote Code Execution: Mitigating shell code injection

- non-executable stack
  - marks the stack memory page as read/write but not executable
  - jump to the stack addresses results in an exception
- stack canaries or cookies
  - random values inserted between return address and called function parameters and stack variables
    - assumption: to overwrite the return address the canary value is also overwritten
  - check the random value when returning (function return is replaced with jump to verification code)
  - problem: array index (integer) overflow techniques can be used to overwrite return address without overwriting the cookie
    - cookie can be overwritten one byte at a time to find out the value of cookie (works in some implementation for `fork`)
- Address Space Layout Randomisations (ASLR)
  - attacker needs to know the address of the injected shell code to jump to it (replace return address with)
    - calculated based on the address of the stack
  - ASLR randomises the process address space
    - program load address, shared libraries, heap and stack base addresses are randomised and will be different with each execution
- Shadow stack
  - return address is also written in a shadow stack
  - the return address is verified at return from function call with value stored in shadow stack

Remote Code Execution: return to library

return to library

- attacker replaces the return address with address of a function in a shared library
  - bypasses the non-executable stack
    - injected value is no longer code and is not executed
    - converts return into a function call

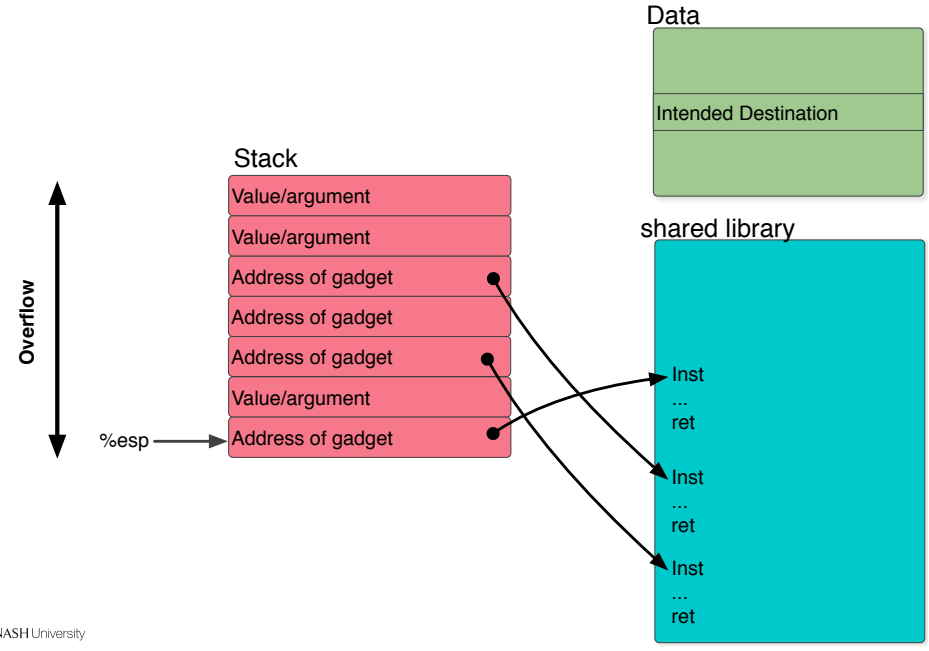


- return address can be an address in the middle of a function in shared library

Remote Code Execution: ROP

- instead of using the complete library function small pieces of code in library are used
  - each piece of code is called a *gadget*
  - a gadget is comprised of few instructions ending with return instruction
    - to get back to the stack that attacker controls
  - each gadget performs a task equivalent to a single machine instruction (in a typical processor)
    - e.g. adding two registers and storing the results
    - however use push and pop (to read from stack and write to stack) to read and write values
- attacker injects addresses and values into the overflown buffer in stack
  - generated by the ROP compiler
- it is shown that a Turing-complete machine (machine capable of executing programs) can be built using these gadgets
  - attacker writes the intended code using high level program
  - a tool compiles the code using gadgets in a specified target library
    - can find the gadgets within the code of a target program

Remote Code Execution: ROP (diagram)



Remote Code Execution: Mitigating ret-to-lib and ROP

- Address Space Layout Randomisations (ASLR)
  - attacker needs to know the address of shared libraries or the executable code itself
    - to inject address of library functions in case of ret-to-lib
    - to inject address of gadgets in case of ROP
- stack canaries/cookies
  - are effective against both
  - can be bypassed by other techniques
- Control Flow Integrity
  - techniques used to check whether jumps and returns follow Control Flow Graph of the program
    - statically generated CFG are prone to attacks (more permissive)
    - dynamically generated CFG are more accurate but more expensive (still active topic in research)
- Shadow stack
  - keeping a duplicate stack is expensive

Malware Incident Prevention

NIST SP 800-83r1 proposes the following elements to prevent malware incidents

- policy
- awareness
- vulnerability mitigation
- threat mitigation
- defensive architecture

Malware Incident Prevention: Policy[1]

- organisation security policy should address malware incident prevention
  - policy statement can then be used to drive the incident prevention efforts
    - IT staff awareness, vulnerability mitigation, defensive architecture
  - without policy the efforts may not be effective and consistent
- Malware prevention policies should be general to be flexible in policy implementation
  - reduce policy update
  - however must be clear in intent and scope
  - should include remote workers
    - including those using hosts outside organisation's control
- Examples of Malware prevention policies:
  - require scanning of media from outside organisation
  - require scanning of email attachment
  - prohibit send/receive of certain file types
  - restrict or prohibit use of unnecessary software
  - restrict use of removable media
  - specify the required type of antivirus software for various types of hosts

[1] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Prevention: Awareness[2]

- explains proper rules of behaviour for use of the organisation IT hosts
- provides guidance to users on malware incident prevention
  - reduces the frequency and severity of incidents
- inform all users of
  - the ways malware enters and infects hosts
  - the risks that malware poses
  - technical controls (security measures) cannot prevent all incidents
- Examples of recommended practices for avoiding malware incident (informed users)
  - not opening suspicious emails from known or unknown entities, clicking on attachments, hyperlinks etc.
  - not clicking on suspicious browser pop-ups
  - not opening files with extensions that are likely be associated with malware
  - not disabling security mechanisms such as antivirus, personal firewall etc.
  - not downloading or executing applications from untrusted sources

[2] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Prevention: Vulnerability Mitigation[3]

- malicious code often exploits existing vulnerability to infect and propagate
- vulnerability mitigation can be achieved by
  - updating OS and applications
  - OS/software reconfiguration to disable vulnerable services
- automated configuration checklist for OS and applications
  - to check the configuration or to apply settings automatically
    - e.g. Microsoft Active Directory Policies
- automated patch management
- use host hardening principles (e.g. NIST SP 800-123: Guide to General Server Security)
  - least privilege
  - disable/remove unneeded services
  - remove unsecured file share
  - remove/change default username and passwords for OS and applications
  - disable automatic execution of binaries and scripts
  - change default file association for files commonly used by malware (e.g. .pif, .vbs)

[3] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Prevention: Threat Mitigation[4]

- is still important even if all vulnerabilities are mitigated
  - e.g. social engineering: trick users to run malware
- antivirus software
  - recommended capabilities
    - scanning critical host components
    - monitor host for suspicious activities
    - monitor applications behaviour
    - file scan for known viruses (less effective against poly/metamorphic malicious code)
    - disinfecting files, quarantine infected files
  - use both host-based and network-based antivirus scanning tools
  - keep antivirus software as well as its signature database up-to-date
  - centrally managed
    - prevent users from disabling/removing the antivirus software
    - automatically update signature database
  - use multiple antivirus software from different vendors for key hosts

[4] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Prevention: Threat Mitigation[5] (continued)

- intrusion prevention system
  - can prevent some network attacks before reaching hosts
  - are customisable allowing to add new attack signatures
  - some IPS can detect anomalous behaviour by monitoring normal network traffic patterns (used as baseline)
  - host-based IPS can monitor a single host using similar principle as network-based IPS
- firewalls
  - deny-by-default policy will block unintentionally open ports (unneeded services)
  - can block a port used by malware exploiting a vulnerable service
  - in case of a major incident firewall can be used to block the traffic at network level
    - or block external sources used by malware to download other malicious code
- Content Filtering/Inspection
  - used to prevent email-based malware propagation
    - spam filtering to prevent spam emails to deliver malware
  - block attachments with types associated with malware and suspicious file extensions
  - used to stop web-based malware threats
    - blacklist or reputation information
  - block browser pop-up windows
- Application Whitelisting
  - provide a list of authorised applications for use on a host

[5] NIST SP 800-83 r1 Guidance on Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Prevention: Defensive Architecture[6]

- used in addition to vulnerability and threat mitigation, to reduce the impact
- BIOS protection
  - protects modification of BIOS
- sandboxing
  - isolate processes or threads within a controlled environment with restricted access to system resources
    - only *safe* operations are allowed within the sandbox
  - can potentially prevent all or some of the malware malicious activity
  - sandbox can be reset to a known-good state at initialisation
- browser separation
  - multiple brands are installed on a single host
    - different browsers are used for different activities (reducing the likelihood of compromised browser be used for sensitive data)
  - allows tighter control e.g. disabling all forms of mobile code for browsers used for corporate tasks
- segregation through virtualization
  - similar to sandboxing in concept however to segregate applications and OS
  - provides a more rigorous segregation
    - e.g. an OS for corporate tasks another for other activities
    - a compromise in one image does not affect the other unless the virtualization software is also compromised
  - images can be reset to a known-good state when restarted

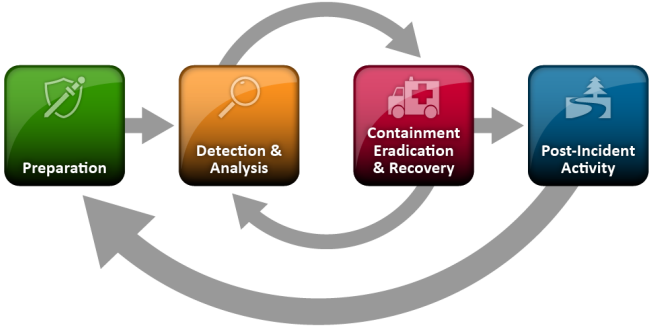
---

[6] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response[7]

- no matter how many protective measures are put in place malware incidents will happen
  - due to unknown types or threats, human error etc.
- must have malware incident response plan to limit the damage and restore data and services
- NIST SP 800-83r1 defines the following life cycle for malware incident response

- Preparation
- Detection and Analysis
- Containment, Eradication, and Recovery
- Post-Incident Activity



[7] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Preparation[8]

- building and maintaining malware-related skills
  - how each major category of malware infects and propagates
  - be familiar with organisation's malware detection tool configuration
  - be able to analyse supporting data
  - be able to identify characteristics of threats
  - have incident handlers temporarily work as antivirus engineers to learn new skills
- facilitating communication and coordination
  - have a coordination team for malware incident response
  - coordinator to provide guidance and instructions to all staff assisting with
    - containment
    - eradication
    - recovery
  - coordinator to update management
  - establish point of contact to answer questions about malware alerts
- acquire tools and resources
  - malware analysis toolkits

[8] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Detection and Analysis[9]

- identifying malware incident characteristics
  - validate malware incident
    - false positives: benign activity as malicious
  - when in doubt treat as a malware incident until confirmed
    - since waiting can have significant impact on the organisation
  - helps with assigning priority to response efforts
  - collaborate with security administrators to identify data source aiding in detection
    - antivirus
    - IDS
    - Security Information and Event Management
  - search for identified characteristics
    - malware category
    - services, ports, protocols attacked
    - exploited vulnerabilities
    - malicious filenames, size, content
    - OS, devices, applications
- identify infected hosts
  - to undergo containment, eradication, and recovery

[9] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Detection and Analysis[10] (continued)

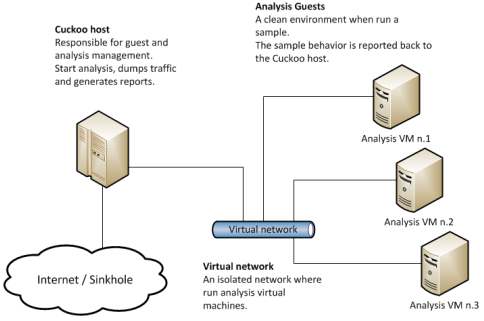
- identify infected hosts ...
  - forensic identification
    - searching for evidence of recent infections
    - security logs, dns server logs, application server logs, network forensic tools
  - active identification
    - security automation: continuous monitoring e.g. network access control technologies
    - custom network-based IDS and IPS: dedicated IDS/IPS with custom signatures for malware detection
  - packet sniffers and protocol analysers
    - configured to look for characteristics of a particular malware
  - manual identification
    - when automated methods are not feasible
- prioritising incident response
  - malware propagation speed e.g. worms
  - value of data when responding to Trojan horse incidents
  - establish a set of criteria to identify the proper level of response for various malicious code
    - type of malware
    - point of entry and propagation method
    - types of attacker tools placed in infected host by malware
    - what hosts and networks are affected
    - how the impact is going to grow if not contained

[10] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Detection and Analysis[11] (continued)

- malware analysis
  - infect and monitor an isolated and clean test host to study behaviour of malware
  - virtualisation can be used where guest OS is isolated
- Example: cuckoo sandbox: open source automated malware analysis tool

- cuckoo host
  - runs the core components of the sandbox
  - manages the entire analysis process
- cuckoo guests
  - physical or virtual machines
  - run malware in an isolated environment
- malware analysis
  - trace malware calls
  - monitor any file system activity of malware
  - dump memory of the malware process
  - monitor network traffic
  - screen shots taken during malware execution
  - memory dump of the machine



Cuckoo automated malware analysis architecture[a]

[a] What is cuckoo?

[11] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Containment[12]

- two major components
  - stop malware propagation
  - prevent further damage to hosts

Containment through **user participation**:

- important in large scale incidents and in non-managed environments
- should have alternate mechanisms to inform users in case email is unavailable
- organisation should not rely (entirely) on this means for containing malware
  - not all users may be able to follow provided instructions successfully
  - users may ignore instructions and carry on performing their usual tasks

[12] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Containment[13] (continued)

Containment through **automated detection**:

- antivirus is the preferred method of automated detection and removal
  - however often antivirus software fail to detect novel malware threats
  - signature updates should be tested before deployment
    - to avoid potential problems introduced with update itself
- content filtering:
  - email servers and clients, anti-spam software etc. configured to
    - block emails or email attachments with certain characteristics
    - only when malware has static characteristics
  - web content filtering
- network based IPS software
  - when deployed inline with malware signature can stop detected packets to reach hosts
  - can prevent incoming and outgoing attempts
  - the effectiveness depends on accuracy of the signature
- executable black-listing
  - OS capability to prevent execution of files named in a list

[13] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Containment[14] (continued)

Containment through **disabling services**:

- when malware use a service extensively, containment may require disabling the service
  - shutting down service used by malware at application layer or network layer
  - service can be disabled at the perimeter of the network
  - portions of a service is disabled e.g. large mailing lists
- a service may provide a channel for infection
- need to understand the consequence of disabling a service
  - for instance the impact on other services
  - should have a list of dependencies between services

[14] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Containment[15] (continued)

Containment through **disabling connectivity**:

- placing restrictions on network connectivity
  - blocking all access of infected hosts to external network
    - prevent download of attacker rootkits
  - block access to local network
    - prevent propagation of malware to other hosts
  - more drastic measures: isolate subnets/groups of hosts
- can design the network to make containment through loss of connectivity easier
  - separate subnets for servers and workstations
    - in malware incidents targeting workstations the workstation subnet can be isolated
  - separation using VLAN
    - separate VLAN for infected hosts
    - NAS servers can place hosts in different VLAN based on security check lists

[15] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Eradication[16]

- primary goal is to remove malware
  - also require eradication of vulnerability used for infection
    - vulnerable service
    - unsecured file share
- antivirus software are most common tools for eradication
  - remote scanning is more effective
  - isolation of infected hosts may make it difficult to remotely access to update
    - placing infected hosts in separate VLAN can provide remote access to incident handlers
- may require user participation
- for may incidents rebuild of all infected hosts are required
  - rebuild and securing OS
  - restoring data from known-good backups
- rebuild any host if:
  - attacker gained administrative level access
  - administrative level access was available to anyone through a backdoor
  - system files were replaced by malware
  - host is unstable
  - doubt about the nature and extent of the infection

[16] T SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Recovery[17]

- restoring functionality and data of infected hosts
- removing temporary containment measures
  - determining when to remove
  - during major incidents may require restoring connectivity while still going through eradication and recovery
    - when vulnerable hosts are patched so restring network connectivity will not lead to new hosts being infected
- should consider worst-case scenarios requiring rebuild of thousands of hosts
  - who will perform the recovery tasks
  - estimating the time required
  - priority of recovery efforts

[17] NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Malware Incident Response: Post-incident Activities

- Changes to security policy
  - to prevent similar incidents
- Changes to awareness program
  - training for users
- Software reconfiguration
  - OS
  - application
- Malware detection software deployment
  - an incident provides justification to deploy malware detection if previously was not used
- Malware detection software reconfiguration
  - increasing signature update frequency
  - increase scope of monitoring
  - change default response to malware detection

References

The materials in this document are reproduced, at times without modification, from the following sources:

- NIST SP 800-83r1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops
- cuckoo sandbox: open source automated malware analysis tool