OPERATING SYSTEMS

PROTECTION AND SECURITY

What is security?

Security as a tradeoff

- Security is about evaluating risks
- Mitigating the risks as an associated cost
- For each application scenario a tradeoff has to be reached between mitigation of the risk, cost of the mitigation and cost of the residual risk

The Value of Things

Cyber Crime

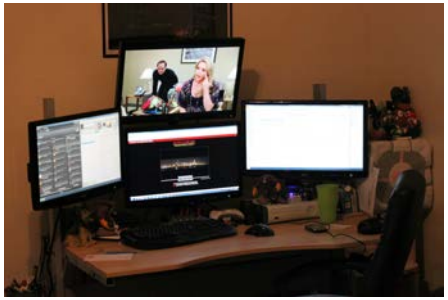
High gain/cost ratio

Goods and Risks are transformed into intangible assets

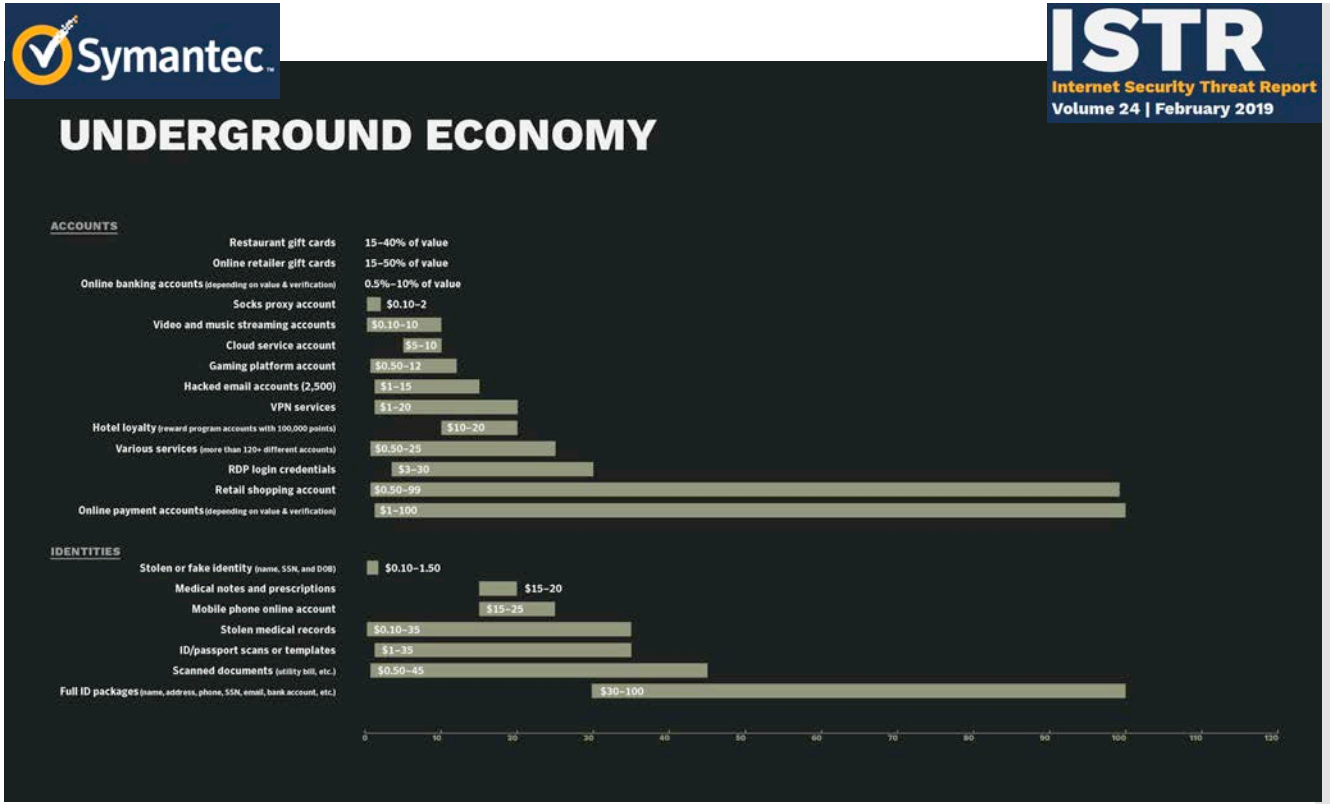
Low material costs

Life is rarely at risk

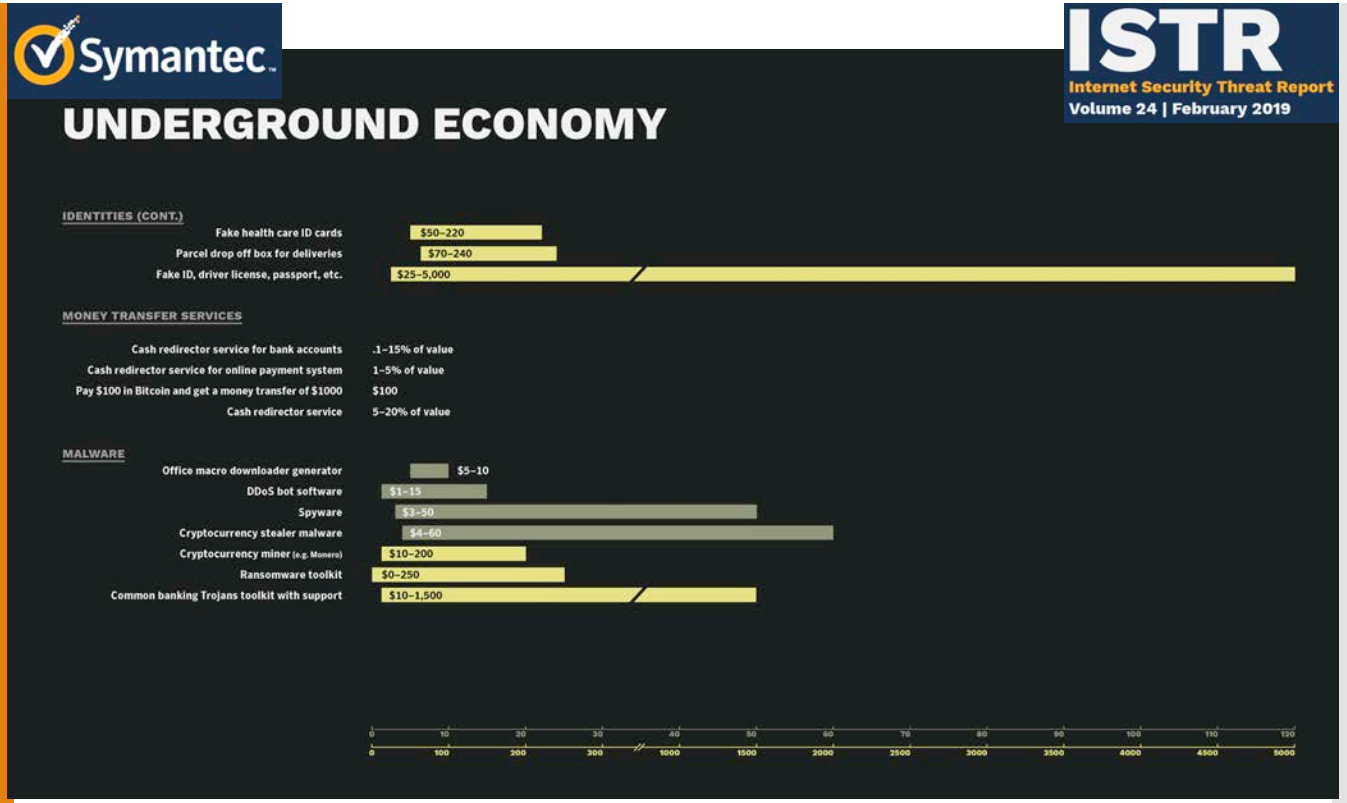
Cyber Crime is often
not perceived as a Crime



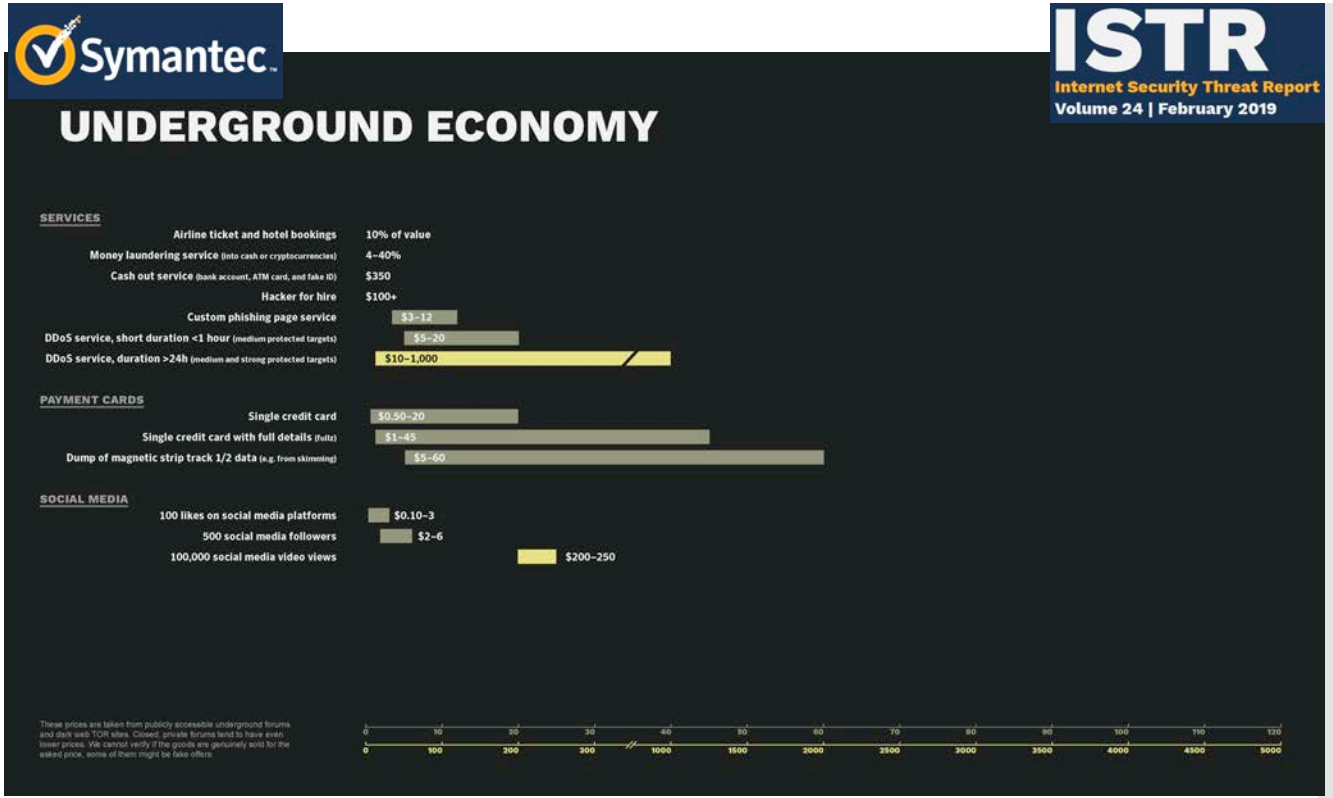
The Economy of CyberCrime



The Economy of CyberCrime



The Economy of CyberCrime



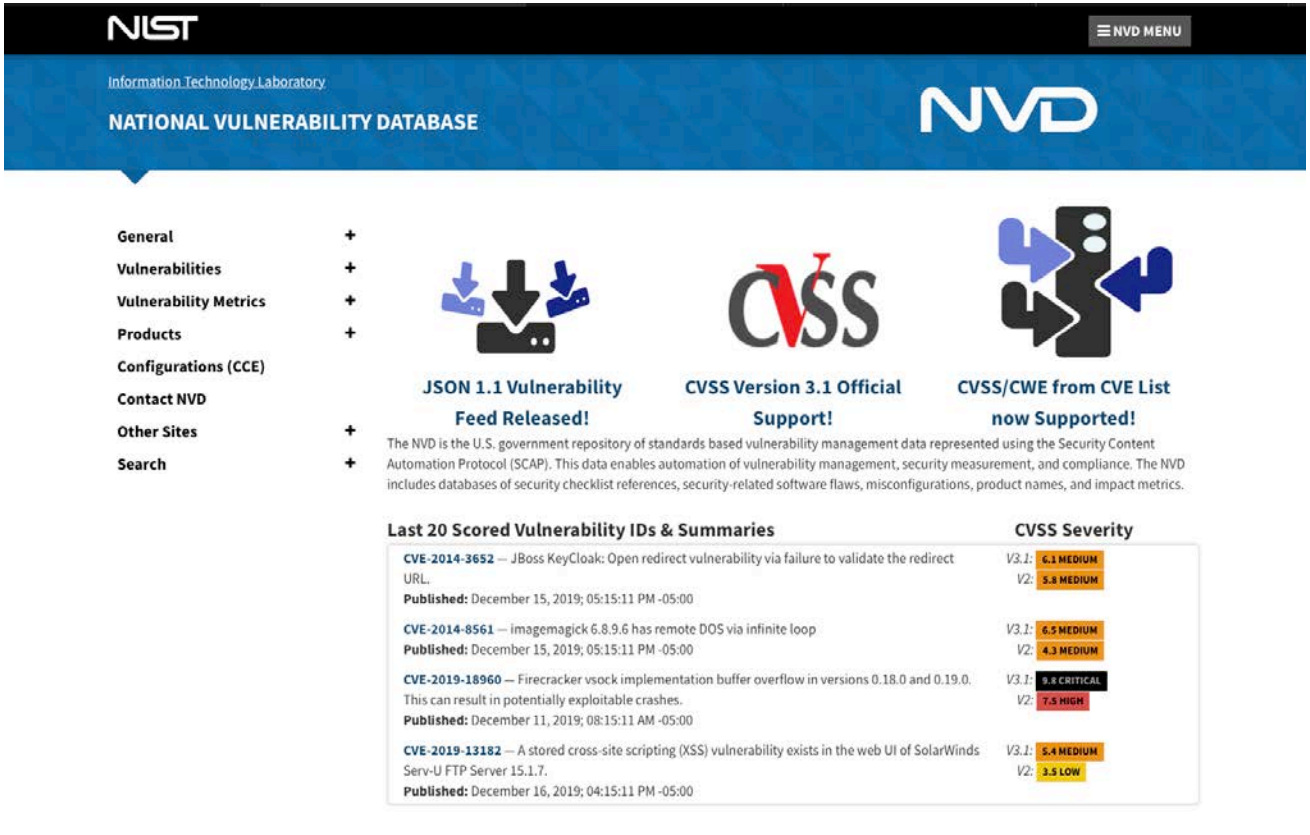
Definitions

- **Vulnerability**
  - Any flaw in the system that can be leveraged to perform attacks against availability, confidentiality and integrity.
    - e.g., lack of access controls, unchecked bounds in C, etc.
- **Threat**
  - The potential for a threat-source to successfully exploit a particular information system vulnerability. (ENISA)
- **Attack**
  - Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself [by exploiting system vulnerabilities] (CNSS)

System and application software

- Any computer program may contain errors or vulnerabilities
  - may depend on the programming language
  - may cause unexpected outputs to unexpected inputs
  - may allow for the modification of the program flow
- A large number of groups analyse program codes to
  - discover vulnerabilities
  - exploit vulnerabilities to perform attacks
- The major software companies, security researchers, and security companies, constantly revise their product to find and correct vulnerabilities

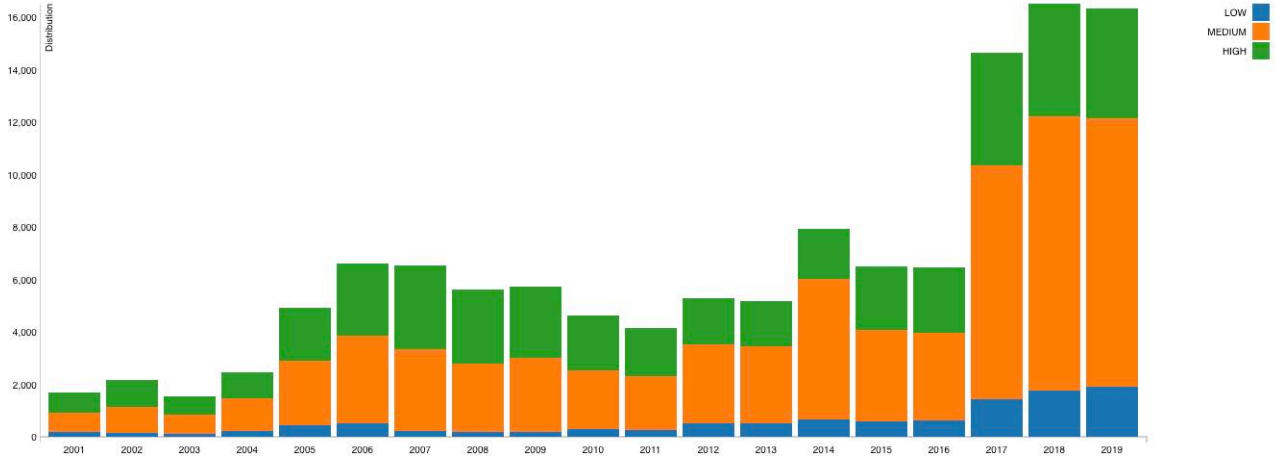
CVE Common Vulnerabilities and Exposures



CVSS Common Vulnerability Scoring System

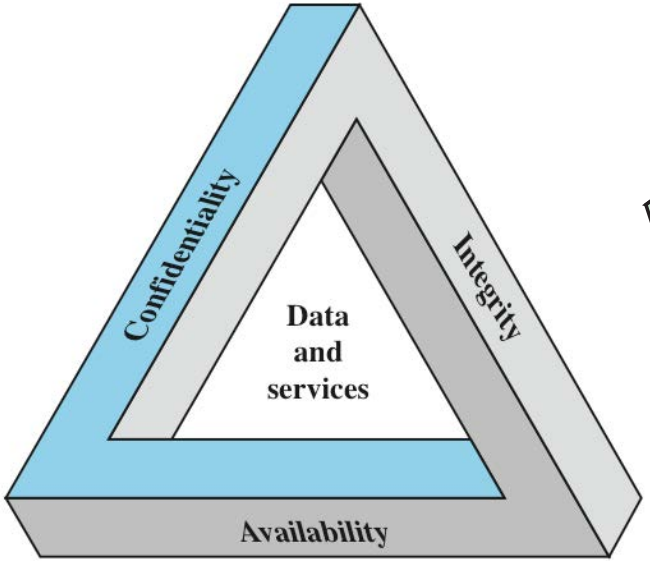
CVSS Severity Distribution Over Time

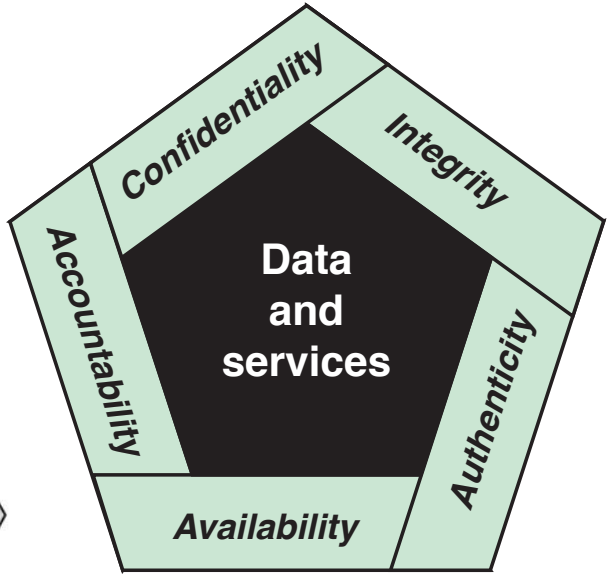
This visualization is a simple graph which shows the distribution of vulnerabilities by severity over time. The choice of LOW, MEDIUM and HIGH is based upon the CVSS V2 Base score. For more information on how this data was constructed please see the NVD CVSS page.



https://nvd.nist.gov/vuln-metrics/visualizations/cvss-severity-distribution-over-time

The CIA Triad

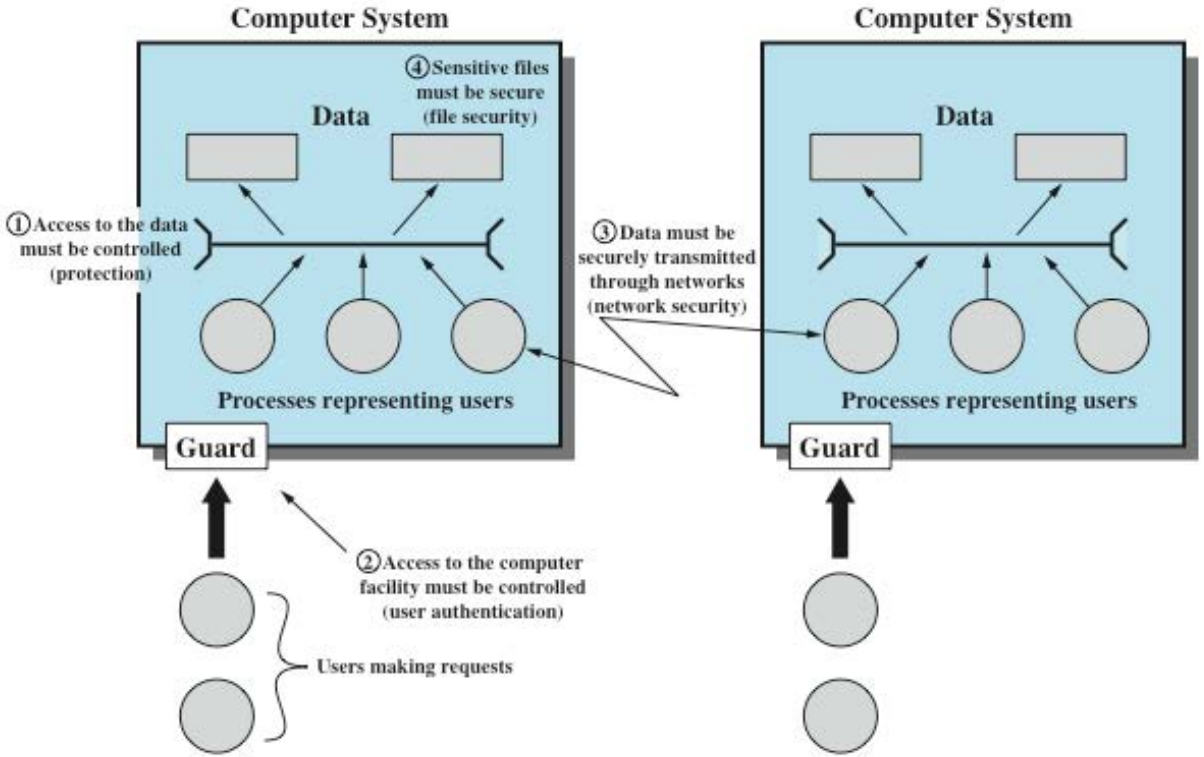




Hardware

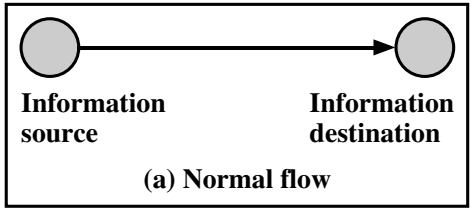
- Availability
  - Damage, steal
  - Power outages
- Confidentiality and Integrity
  - access to memory, register locations
  - access to communication lines

Architecture of a Computer Systems from a Security Perspective



Threat Model

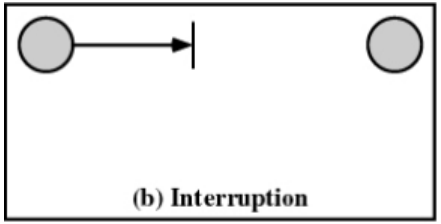
- Any action performed by a computer system can be modelled as an information flow from a source to a sink



- Computer attacks aim at modifying the information flow
- Four main categories of attacks can be defined

1. Interruption

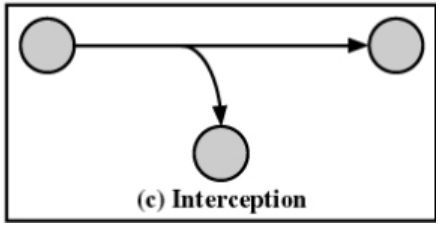
- An asset is destroyed or disabled
  - hardware damages
  - interruption of communication lines
  - exhausting all the available resources
  - disabling core services



- This kind of attack is called Denial of Service (DoS) as the attack threats the availability

2. Interception

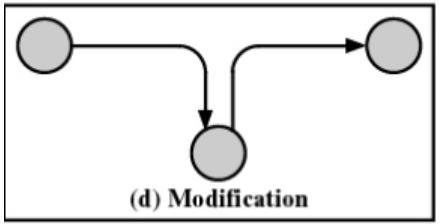
- A third unauthorised party gain access to information flows



- This attack is a threat to confidentiality

3. Modification

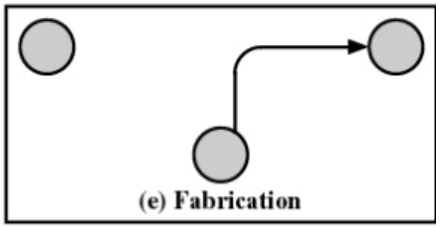
- A third unauthorised party
  - intercepts the information flow by *spoofing* the identity of the destination (this is an attack per se)
  - sends a *modified* flow to the destination



- This attack is a threat to confidentiality and integrity

4. Fabrication

- A third unauthorised party produces information flows by *spoofing* the identity of the source



- This attack is a threat to integrity

Threats

System Access Threats

System access threats fall into two general categories

Intruders

Malicious software

Intruders

Masquerader

An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account

Misfeasor

A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges

Clandestine user

An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

Means of Authentication

- Something the individual knows
  - Examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions
- Something the individual possesses
  - Referred to as a token. Examples include electronic keycards, smart cards, and physical keys
- Something the individual is (static biometrics)
  - Examples include recognition by fingerprint, retina, and face
- Something the individual does (dynamic biometrics)
  - Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm
- Multifactor authentication: two or more different *types*

Principles of Protection

- Principle of least privilege
  - Programs, users and systems should be given just enough privileges to perform their tasks
- Properly set permissions can limit damage if entity has a bug, gets abused
  - Can be static (during life of system, during life of process) or dynamic (changed by process as needed)
- Compartmentalization
  - Process of protecting each individual system component through the use of specific permissions and access restrictions

Protection Rings

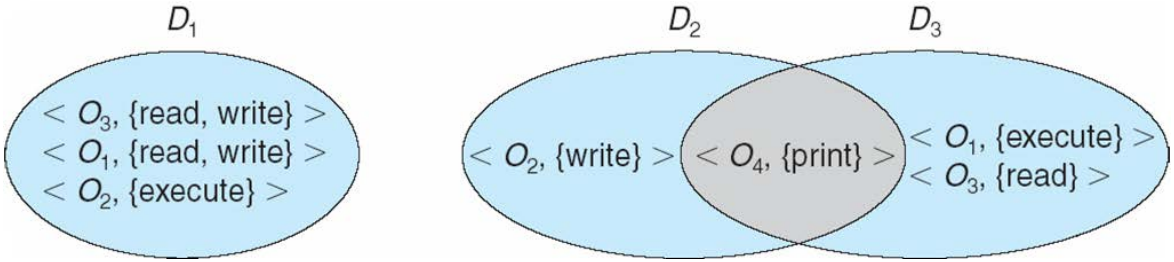
- Components ordered by amount of privilege and protected from each other
  - For example, the kernel is in one ring and user applications in another
- Privilege separation requires hardware support
- Hypervisors introduced the need for yet another ring

Domain of Protection

- Rings of protection separate functions into domains and order them hierarchically
- Computer can be treated as processes and objects
  - Hardware objects (such as devices) and software objects (such as files, programs, semaphores
- A process should only have access to objects it currently requires to complete its task – the need-to-know principle
- Implementation can be via process operating in a protection domain
  - Each domain specifies set of objects and types of operations on them
  - Ability to execute an operation on an object is an access right <object-name, rights-set>

Domain Structure

- Access-right = <object-name, rights-set>
  where rights-set is a subset of all valid operations that can be performed on the object
- Domain = set of access-rights



Domain Implementation (UNIX)

- Domain = user-id
- Domain switch accomplished via file system
  - Each file has associated with it a domain bit (`setuid` bit)
  - When file is executed and `setuid` = on, then user-id is set to owner of the file being executed
  - When execution completes user-id is reset
- Domain switch accomplished via passwords
  - `su` command temporarily switches to another user's domain when other domain's password provided
- Domain switching via commands
  - `sudo` command prefix executes specified command in another domain

Domain Implementation (Android App IDs)

- In Android, distinct user IDs are provided on a per-application basis
- When an application is installed, the installd daemon assigns it a distinct user ID (UID) and group ID (GID), along with a private data directory (/data/data/<appname>) whose ownership is granted to this UID/GID combination alone.
- Applications on the device enjoy the same level of protection provided by UNIX systems to separate users.

Access Matrix

- View protection as a matrix
  - Rows represent domains
  - Columns represent objects
- Access(i, j) is the set of operations that a process executing in $Domain_i$ can invoke on $Object_j$

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Use of Access Matrix

- If a process in Domain D_i tries to do "op" on object O_j, then "op" must be in the access matrix
- The user who creates an object can define the access column for that object
- Can be expanded to dynamic protection
- Operations to add, delete access rights

Implementation of Access Matrix

- Generally, a sparse matrix
- Option 1 – Global table
  - Store ordered triples <domain, object, rights-set> in table
  - A requested operation M on object O_j within domain D_i -> search table for $< D_i, O_j, R_k >$
  - Table could be too large to fit in main memory
  - Difficult to group objects

Implementation of Access Matrix

- Option 2 – Access lists for objects
  - Each column implemented as an access list for one object
  - Resulting per-object list consists of ordered pairs <domain, rights-set> defining all domains with non-empty set of access rights for the object

Implementation of Access Matrix

- Option 3 – Capability list for domains
  - Capability list for domain is list of objects together with operations allowed on them
  - Execute operation M on object O_j, process requests operation and specifies capability as parameter
  - Capability list associated with domain but never directly accessible by domain

Implementation of Access Matrix

- Option 4 – Lock-key
  - Compromise between access lists and capability lists
  - Each object has list of unique bit patterns, called locks
  - Each domain as list of unique bit patterns called keys
  - Process in a domain can only access object if domain has key that matches one of the locks

Comparison of implementations

- Global table is simple, but can be large
- Access lists correspond to needs of users
- Capability lists useful for localizing information for a given process
- Lock-key effective and flexible, keys can be passed freely from domain to domain, easy revocation

Other Protection Improvement Methods

- System integrity protection (SIP)
  - Introduced by Apple in macOS 10.11
  - Restricts access to system files and resources, even by root
  - Uses extended file attributes to mark a binary to restrict changes, disable debugging and scrutinizing
  - Also, only code-signed kernel extensions and only code-signed apps
- System-call filtering

Other Protection Improvement Methods

- Sandboxing
  - Running process in limited environment
  - Impose set of irremovable restrictions early in startup of process (before `main()`)
  - Process then unable to access any resources beyond its allowed set
  - Java and .NET implement at a virtual machine level

Other Protection Improvement Methods

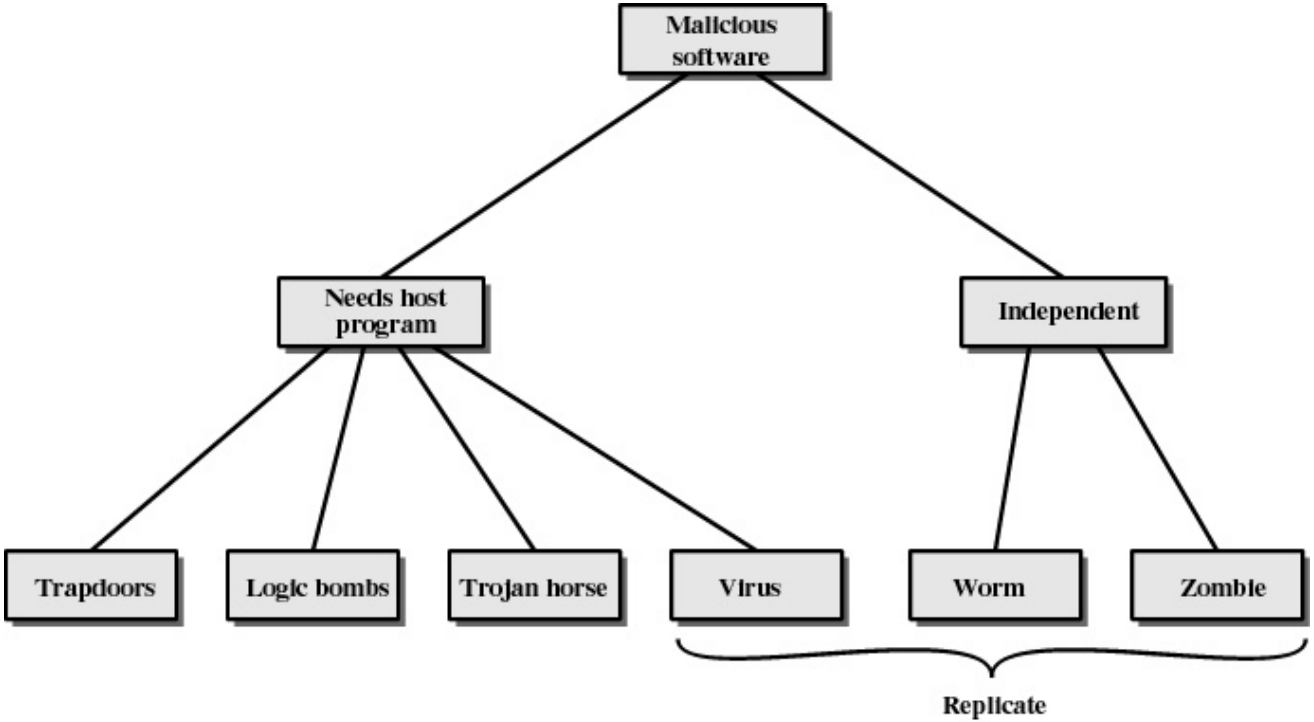
- Code signing allows a system to trust a program or script by using crypto hash to have the developer sign the executable
  - If the code is changed, signature invalid and (some) systems disable execution
  - Can also be used to disable old programs by the operating system vendor (such as Apple) cosigning apps, and then invaliding those signatures so the code will no longer run

Malware

Malicious Software

- Programs that exploit vulnerabilities in computing systems
- Also referred to as malware
- Can be divided into two categories:
  - Parasitic
    - Fragments of programs that cannot exist independently of some actual application program, utility, or system program
    - Viruses, logic bombs, and backdoors are examples
  - Independent
    - Self-contained programs that can be scheduled and run by the operating system
    - Worms and bot programs are examples

Basic Malware Taxonomy



Buffer Overflow Attacks

- Also known as a buffer overrun
- Defined in the NIST (National Institute of Standards and Technology) Glossary of Key Information Security Terms as:

> *"A condition at an interface under which more input can be placed into a buffer or data-holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system"*

Basic Buffer Overflow Example

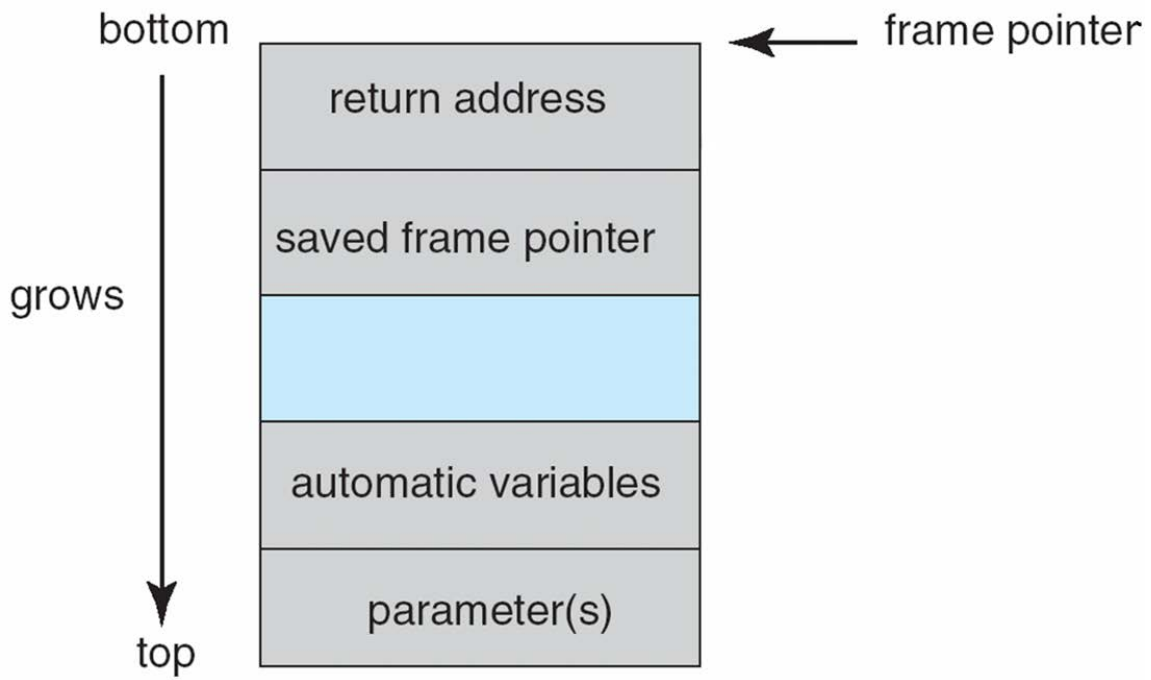
```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

(a) Basic buffer overflow C code

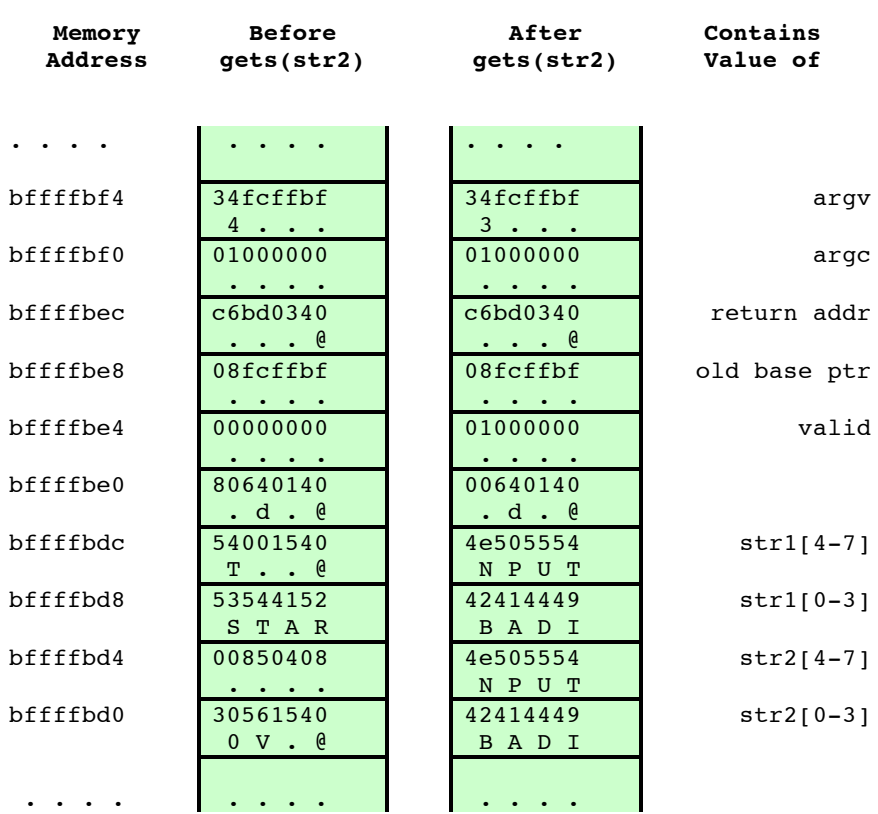
```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

(b) Basic buffer overflow example runs

Stack configuration



Basic Buffer Overflow Example Stack Values



Language-Based Protection

- Specification of protection in a programming language allows the high-level description of policies for the allocation and use of resources
- Language implementation can provide software for protection enforcement when automatic hardware-supported checking is unavailable
- Interpret protection specifications to generate calls on whatever protection system is provided by the hardware and the operating system

Compile-time Techniques

- Choice of programming language
  - Using a high-level programming language that has a strong notion of variable type and what constitutes permissible operations on them
  - The flexibility and safety provided by these languages does come at a cost in resource use, both at compile time and also in additional code that must execute at runtime
- Language extensions and use of safe libraries
- Safe coding techniques
- Stack protection mechanisms
  - For example, by instrumenting the function entry and exit code to set up and then check its stack frame for any evidence of corruption

Runtime Techniques

- Executable address space protection
  - DEP – Data Execution Prevention
- Address Space Layout Randomization (ASLR)
  - Manipulation of the location of key data structures in the address space of a process
  - Randomizing the order of loading standard libraries by a program and their virtual memory address locations
- Guard pages
  - Caps are placed between the ranges of addresses used for each of the components of the address space
  - These gaps, or guard pages, are flagged in the MMU as illegal addresses and any attempt to access them results in the process being aborted

OS Hardening

Operating Systems Hardening

- Install and patch the operating system
- Configure the operating system to adequately address the identified security needs of the system by
  - Removing unnecessary services, applications, and protocols
  - Configuring users, groups and permissions
  - Configuring resource controls
- Install and configure additional security controls, such as antivirus, host-based firewalls, and intrusion detection systems (IDS), if needed

Security Maintenance

Monitoring and analyzing logging information

Performing regular backups

Recovering from security compromises

Regularly testing system security

Using appropriate software maintenance processes to patch and update all critical software and to monitor and revise configuration as needed

Logging

- Logging can generate significant volumes of information: space should be allocated for them
- Automatic log rotation and archive system allows managing the overall size of the logging information
- Some form of automated analysis is preferred as it is more likely to identify abnormal activity

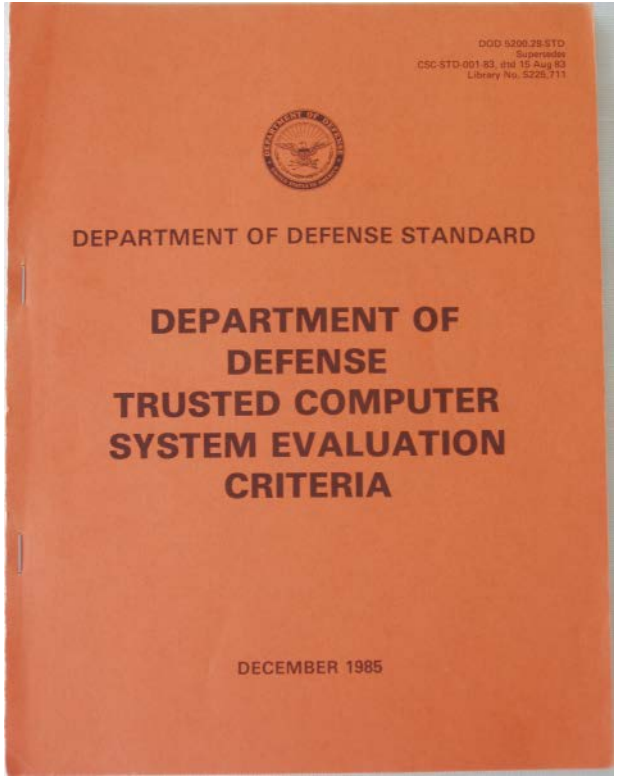
Data Backup and Archive

- Backup
  - Making copies of data at regular intervals, allowing the recovery of lost or corrupted data over relatively short time periods of a few hours to some weeks
- Archive
  - Retaining copies of data over extended periods of time in order to meet legal and operational requirements to access past data
- Key decisions include
  - whether the copies should be kept online or offline
  - whether copies should be stored locally or transported to a remote site

Certifications

Orange Book (1985)

- First document on software certification: the "Orange Book" – US Department of Defense (DoD). Currently available at
  - http://www.dynamoo.com/orange
  - https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/dod85.pdf
- Criteria to evaluate the *security* of operating systems, categorized into seven classes
- Outdated categorization, still valid principles



Orange Book categorization (1)

- **D**: Minimal Protection
  - OSs that fail to meet the requirements for a higher evaluation class
    - MS-DOS, Windows 95/98/ME
- **C**: Discretionary Protection
  - the administrator **can** apply protection mechanisms to objects
  - the OS provides some basic logging capabilities
    - **C1 – Discretionary Security Protection**: users–data separation
      early UNIX versions
    - **C2 – Controlled Access Protection**: fine-grained access control
      IBM OS/400, Win NT/2000/XP, Novell Netware

Orange Book categorization (2)

- **B**: Mandatory Protection
  - the OS requires to assign protection levels to each object
    - **B1 – Labeled Security Protection** (e.g., process, file, device)
      HP-UX, Cray Research Trusted Unicos 8.0, Digital SEVMS
    - **B2 – Structured Protection**: formal security policy model
      Honeywell Multics, Cryptek VSLAN, trusted XENIX
    - **B3 – Security Domains:** mediation of accesses of subjects to objects
      Getronics/Wang Federal XTS-300
- **A** (A1): Verified Protection
  - trustworthiness of the OS is verified through formal methods
    - Boeing MSL LAN, Honeywell SCOMP

Common Criteria (CC)

- A common set of criteria for evaluating the security of computer systems, developed by the national security authorities of USA, Canada and Europe
  https://www.commoncriteriaportal.org

- Common Criteria Recognition Agreement (CCRA)
  - product evaluation by **independent**, **licensed** laboratories
  - documents defining the certification process
  - certifications issued by **Certificate Authorizing Schemes** (subset of CCRA members)
  - certifications recognized by all CCRA members

CC examples of certified products

EAL 7+

- Fort Fox Hardware Data Diode, versie FFHDD2+

EAL 7

- Virtual Machine of Multos M3 G230M mask with AMD 113v4
- Memory Management Unit des microcontrôleurs SAMSUNG S3FT9KF/ S3FT9KT/ S3FT9KS en révision 1

EAL6+

- Green Hills Software INTEGRITY-178B Separation Kernel, comprising: INTEGRITY-178B Real Time Operating System (RTOS), version IN-ICR750-0101-GH01_Rel running on Compact PCI card, version CPN 944-2021-021 with PowerPC, version 750CXe
- Infineon Security Controller M7893 B11 with optional RSA2048/4096 v1.03.006, EC v1.03.006, SHA-2 v1.01 libraries and Toolbox v1.03.006 and with specific IC dedicated software (firmware)

CC examples of certified products

EAL4+

- Red Hat Enterprise Linux Version 7.1
- SUSE Linux Enterprise Server Version 12
- JBoss Enterprise Application Platform 6 Version 6.2.2
- Microsoft SQL Server 2014 Database Engine Enterprise Edition x64
- FINX RTOS Security Enhanced (SE) v3.1

EAL4

- Microsoft Windows 10 Anniversary Update Home Edition, Pro Edition and Enterprise Edition (32 and 64 bits), and Microsoft Windows Server 2016 Standard Edition and Datacenter Edition
- IBM z/OS Version 2 Release 1