Bezpečnosť OS, Ochrana voči útokom

Úvod do informačnej bezpečnosti

Michal Rjaško

Obsah

- Bezpečnosť OS
- Intrusion detection and prevention systems, Antiviruses
- Denial of service útoky
- Penetračné testovanie

OS principles

- hardware abstraction
- resource management: accounting, scheduling, and synchronisation
- storage and communication services: file systems, network, inter-process communication (IPC)
- libraries of common functions: libc
- management of user interaction and interface

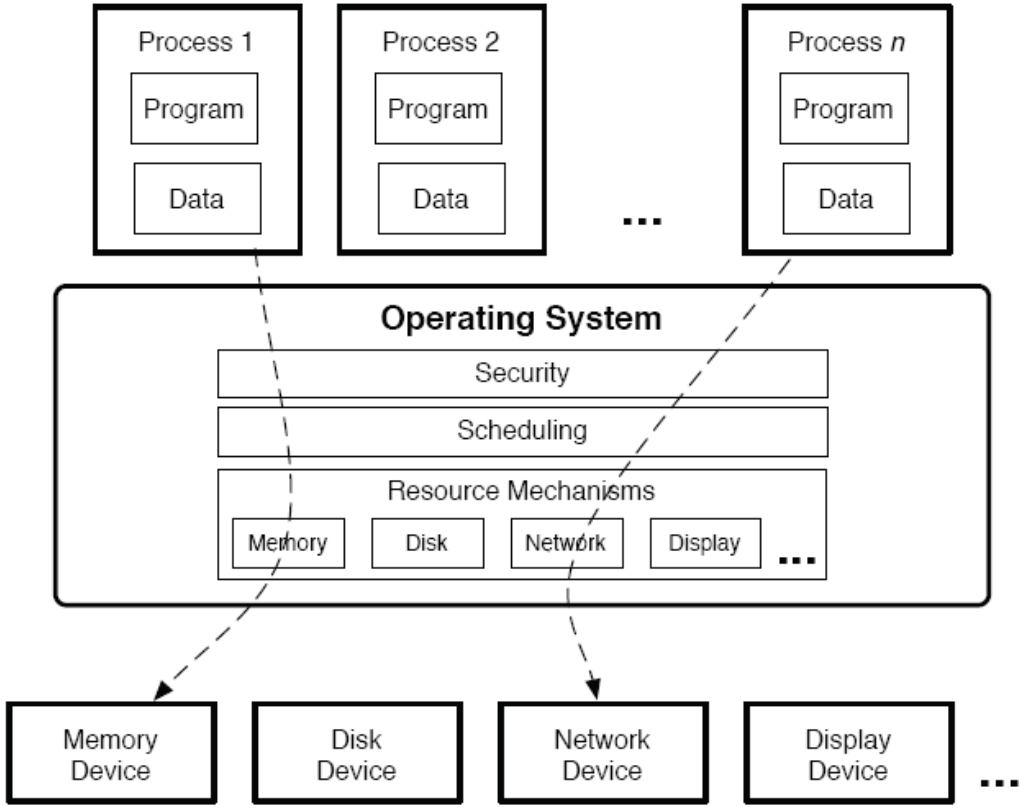


Figure 1.1: An operating system runs *security*, *scheduling*, and *resource mechanisms* to provide *processes* with access to the computer system's resources (e.g., CPU, memory, and devices).

OS security

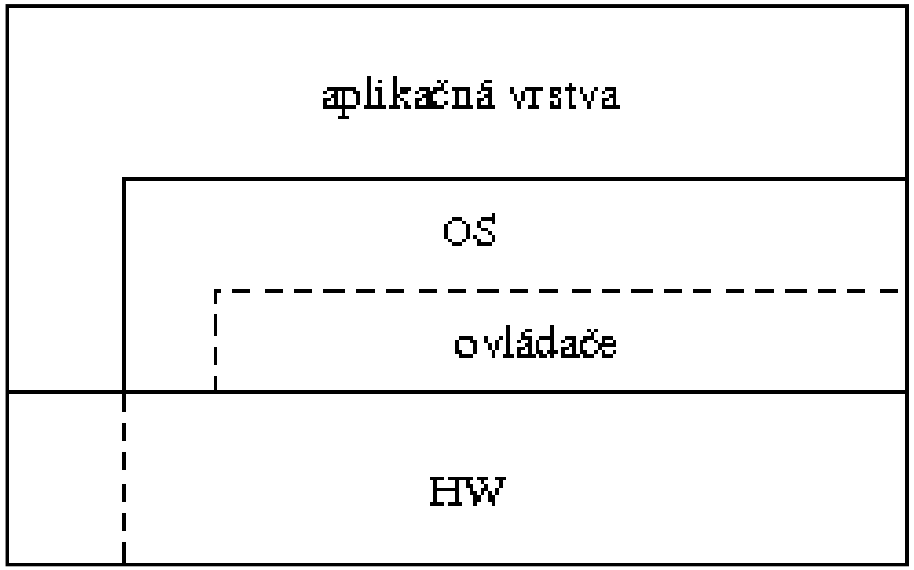
- Operating systems provide the fundamental mechanisms for securing computer processing

What should the OS protect?

- Itself (from users, against motivated adversary)
- Processes (both services and user's application)
- Files access
- Communication (both IPC and network)

Vrstvy informačného systému

- Aplikačná vrstva
  - Klient
  - Server
- (databázová vrstva)
- Operačný systém
- Hardvér
- Prostredie
  - Sieť
  - Priestory, ľudia



Funkcie vrstiev

- Poskytovanie služieb vyšším vrstvám
  - abstrakcia nižších vrstiev
- Izolácia vyšších vrstiev od nižších
  - dôležitý predpoklad pre účinnú implementáciu bezpečnostných mechanizmov
  - nie vždy úplná
    - napr. aplikácia priamo používa časť CPU
- Bezpečnostné mechanizmy je potrebné implementovať na každej vrstve
  - Ťažko sa chráni proti útokom zdola
  - Dole sa ťažko formulujú požiadavky zhora
    - pre procesor sú dáta dáta – pre aplikáciu majú dáta štruktúru a sémantiku (význam)

Bezpečnostné mechanizmy vrstiev

- aplikačná vrstva
  - Ochrana proti útokom prostredníctvom aplikácie
  - Nepomôže proti útokom na úrovni OS
    - Crypto môže čiastočne zabrániť / sťažiť útokom zdola
- hardvér
  - príliš nízko pre aplikačné bezpečnostné problémy
  - nutné pre ochranu OS
  - (napr. executable / non-executable pamäť)
- OS
  - ochrana dát (súborov)
  - ochrana aplikácií (pred inými aplikáciami)

Bezpečnostné funkcie OS

- Identifikácia a autentifikácia používateľov
- Separácia aplikácií
  - obmedzenie vzájomného ovplyvňovania sa
- Riadenie prístupu k prostriedkom
  - voliteľné riadenie prístupu (discretionary access control, DAC)
  - povinné riadenie prístupu (manadatory access control, MAC)
- Ochrana systému a hardvéru

Identifikácia a autentifikácia

- Mali sme o tom prednášku

- Najčastejšie sa používajú heslá
  - Uložené zahašované s využitím náhodnej soli
- V súčasnosti sa rozširuje použitie biometriky
  - Rozpoznanie tváre, dúhovky, odtlačky prstov, ...
- Tokeny
  - Zvyčajne v kombinácii s PINom
  - Napr. elektronický občiansky preukaz

Separácia aplikácií

- oddelenie aplikácií
  - samostatné adresné priestory
  - medziprocesová komunikácia
    - zdieľaná pamäť, semafóry, posielanie správ
  - ladiace (debugging) nástroje
- ochrana systému a hardvéru
  - prístup k OS len cez systémové volania
  - privilegované inštrukcie
  - ochrana prístupu do pamäte

Voliteľné riadenie prístupu

- objekty majú vlastníkov
- vlastníci určujú prístupové práva

- bežné v mnohých OS
  - UNIX
  - Windows

- nedostatočné pre ochranu pred inými aplikáciami rovnakého používateľa

Povinné riadenie prístupu

- prístupové práva určené politikou
  - bežné programy ju nemôžu ovplyvniť
- známe zo sveta utajovaných skutočností
  - Bell-LaPadula model – dôvernosť
  - Biba model – integrita
  - objekty a subjekty majú bezpečnostnú úroveň a množinu kategórií
  - $(u_1, M_1) \leq (u_2, M_2) \Leftrightarrow u_1 \leq u_2 \wedge M_1 \subseteq M_2$

Povinné riadenie prístupu

- Bell – LaPadula model
  - subjekt S môže čítať z objektu O ak
    $(u_O, M_O) \leq (u_S, M_S)$ (t.j. zdola)
  - subjekt S môže zapisovať do objektu O ak
    $(u_S, M_S) \leq (u_O, M_O)$ (t.j. hore)
  - zabraňuje „úniku“ tajnejšej informácie do menej tajného objektu
  - dôveryhodné subjekty môžu písať aj „dolu“
    - môžu informáciu „odtajniť“

Povinné riadenie prístupu

- Biba model
  - subjekt S môže čítať z objektu O ak
    $(u_S, M_S) \leq (u_O, M_O)$ (t.j. zhora)
  - subjekt S môže zapisovať do objektu O ak
    $(u_O, M_O) \leq (u_S, M_S)$ (t.j. dolu)
  - zabraňuje ovplyvneniu dôveryhodnejšieho objektu informáciou z nedôveryhodnejšieho objektu
  - dôveryhodné subjekty môžu písať aj „hore“
    - môžu informáciu „zdôveryhodniť“

Povinné riadenie prístupu

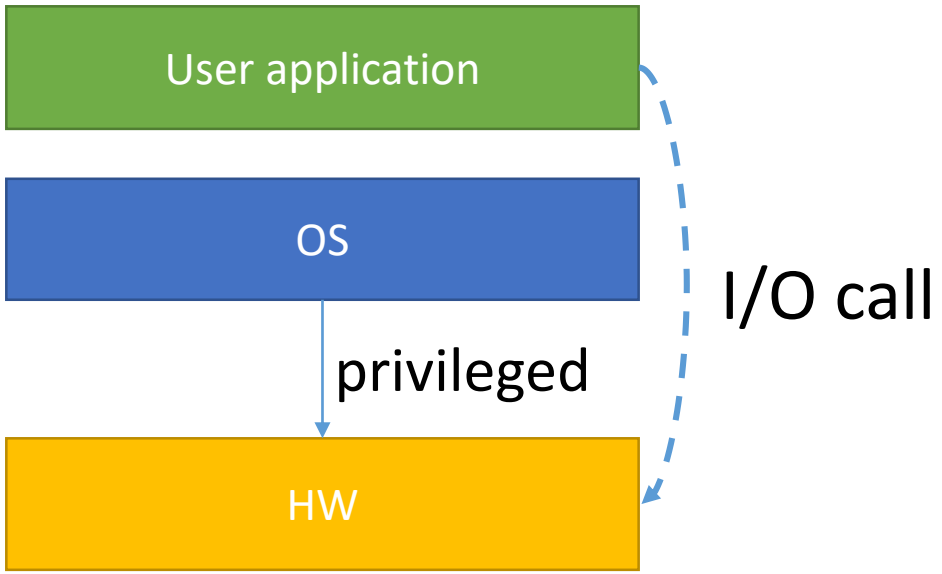
- Použiteľnosť Bell-LaPadula a Biba modelov
  - vytvorené pre utajované skutočnosti (Bell-LaPadula)
- V bežnom prostredí problémy
  - príliš veľa informačných tokov zakázaným smerom
  - príliš veľa subjektov musí byť **dôveryhodných**
  - hrubá granularita dôveryhodnosti subjektov
  - zaslúžia si subjekty „dôveryhodnosť“ ?
    - žiaľ, veľmi nie

Povinné riadenie prístupu

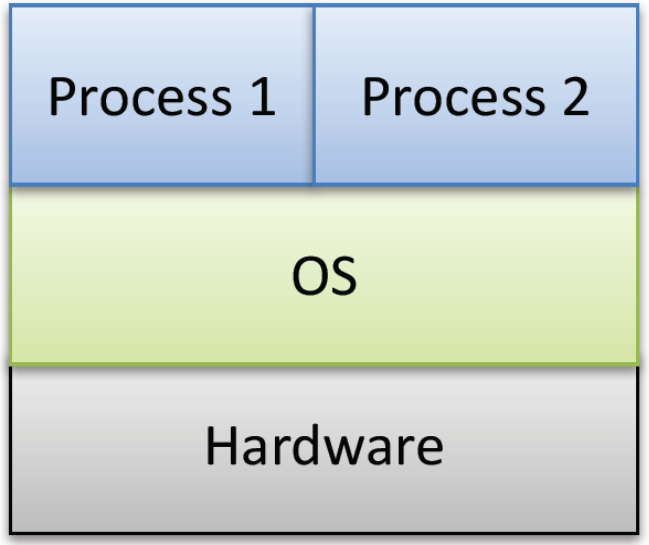
- Domain and Type enforcement (DTE)
  - subjekty pracujú v **doméne**, objekty majú **typ**
  - politika určuje
    - povolené operácie pre doménu a typ
    - prechody medzi doménami
    - typy nových objektov na základe domény subjektu a typu „rodičovského“ objektu
- SELinux
- veľmi flexibilné
  - napr. umožňuje aj implementáciu Bell-LaPadula a Biba modelov

Virtualization

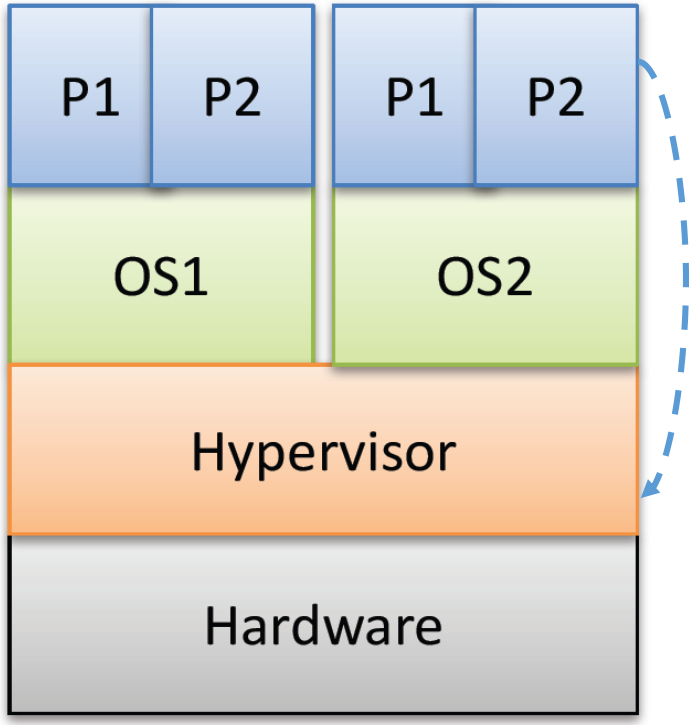
- HW assisted
  - Operating-system thinks it is running on and interacting with its own hardware
  - Abstracts the hardware peripherals from the operating-system
- Operating-System level
  - Makes the subsystem thinks it is running in its own operating-system
  - Abstracts the services and kernel from an application



Virtualization (HW assisted)



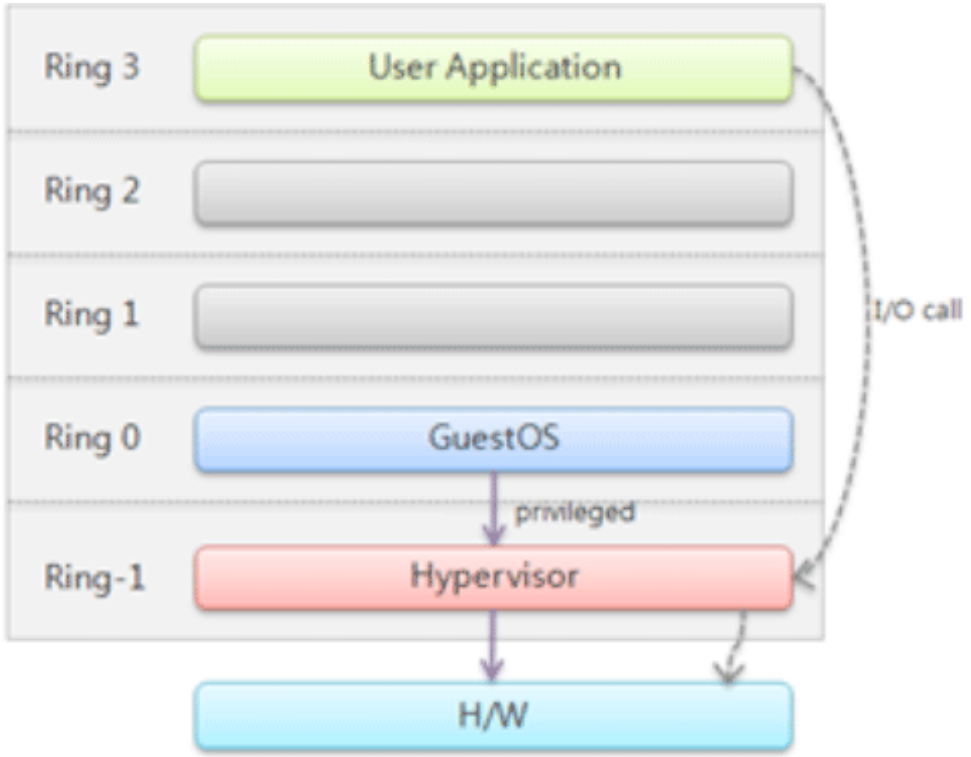
No virtualization

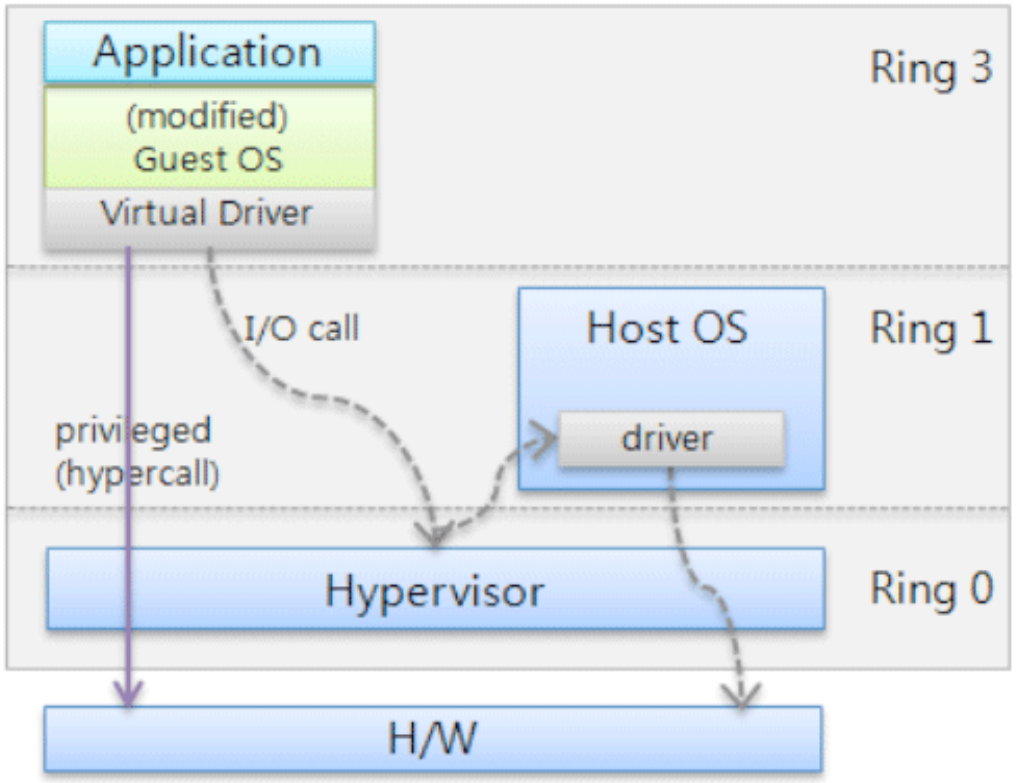


Full virtualization

- Hypervisor may run directly on hardware

Virtualization (HW assisted)

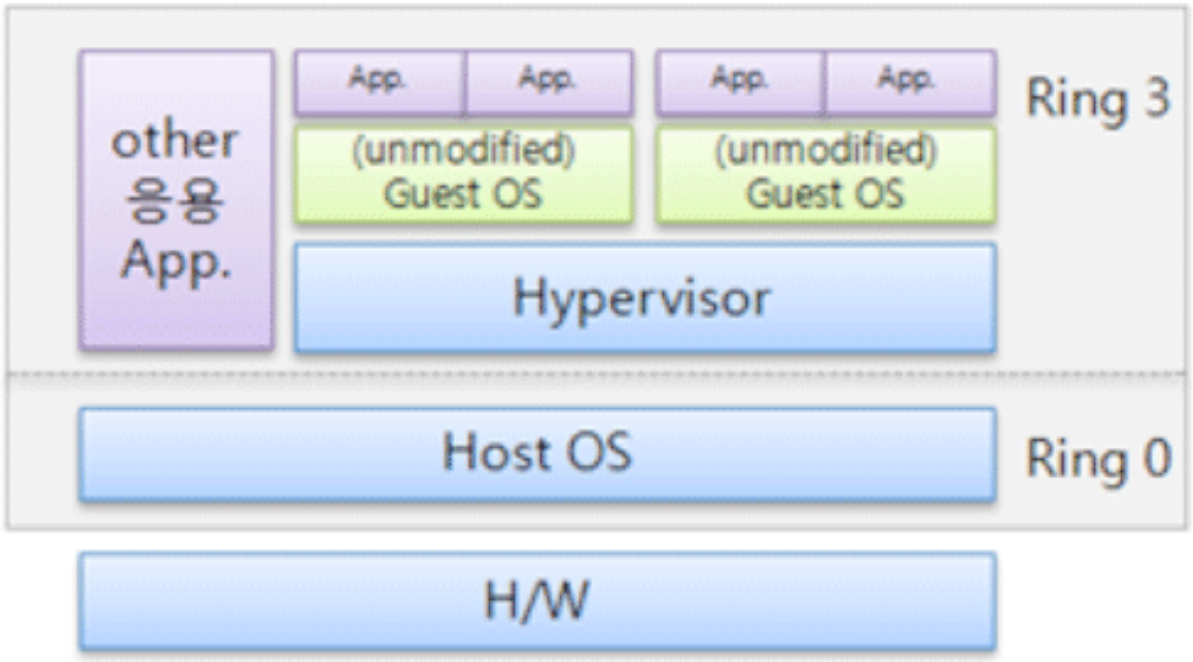




Paravirtualization

- Paravirtualization: hypervisor provides an API and the OS of the guest virtual machine calls that API - requires OS modifications

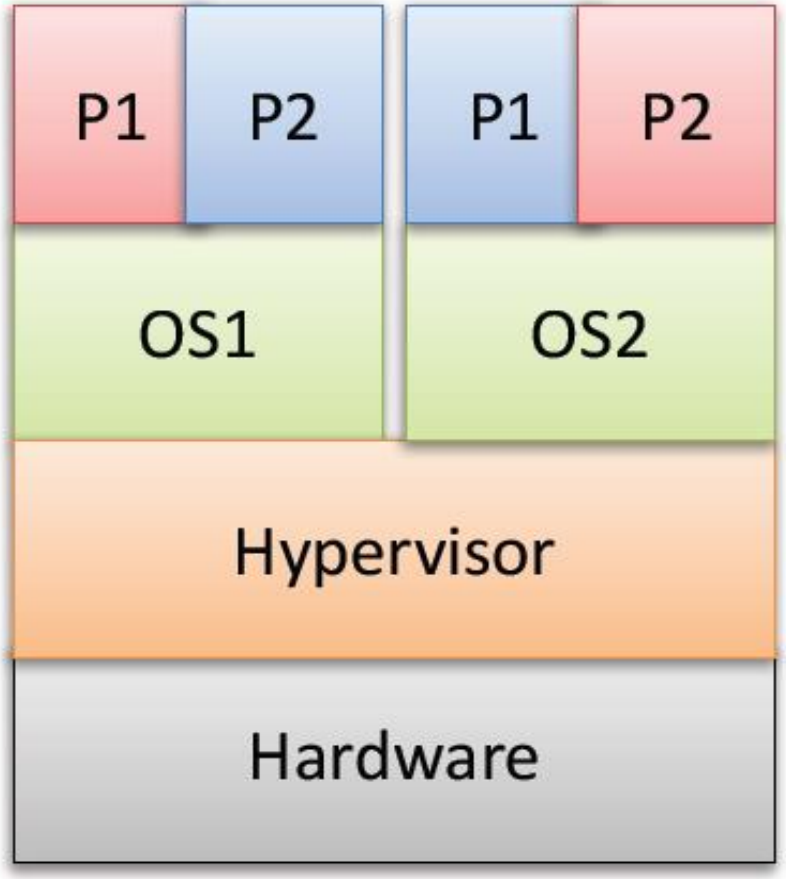
Virtualization (Host - based)



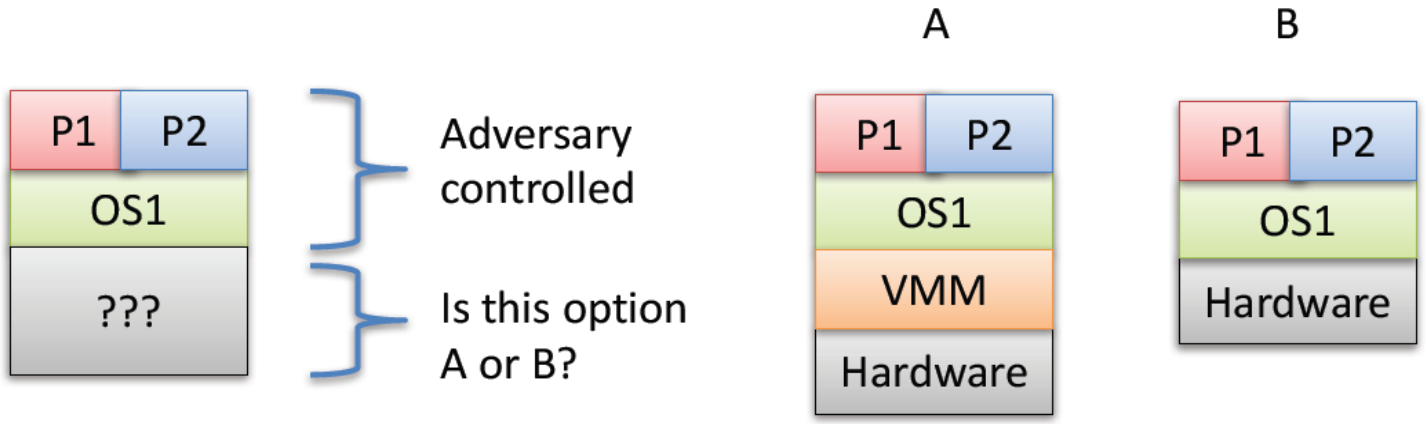
- Relies on an underlying operating-system
  - security environment is less controlled
- There is no direct access to hardware
  - Increased resource overhead of the guest OS

Virtualization use case – study of malware

- Researchers use VMs to study malware
- Example of VM sandboxing
  - Hypervisor must contain malicious code
- How would you evade analysis as a malware writer?



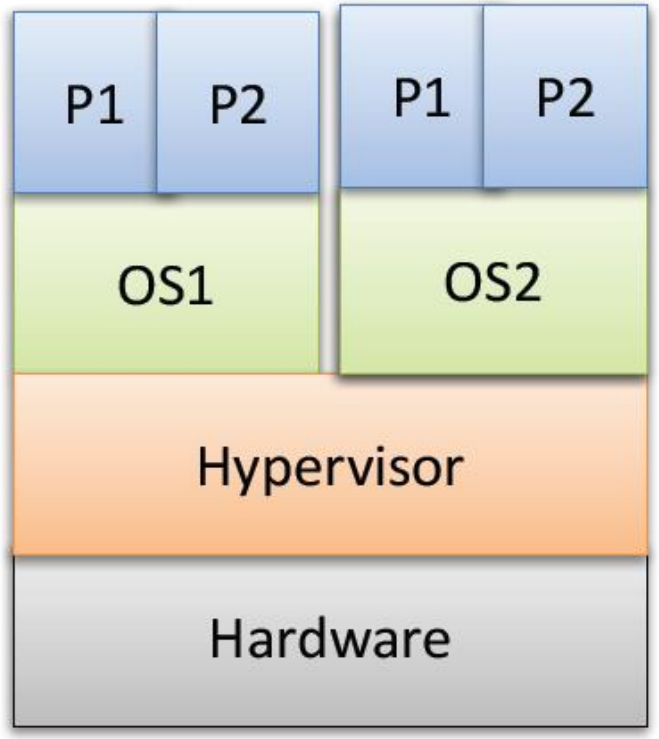
VM transparency



- Adversary can detect if:
  - Paravirtualization
  - Logical discrepancies
    - Expected CPU behavior vs virtualized
    - Red pill (Store Interrupt Descriptor Table http://www.securiteam.com/securityreviews/6Z00H20BQS.html)
- Timing discrepancies
  - Slower use of some resources

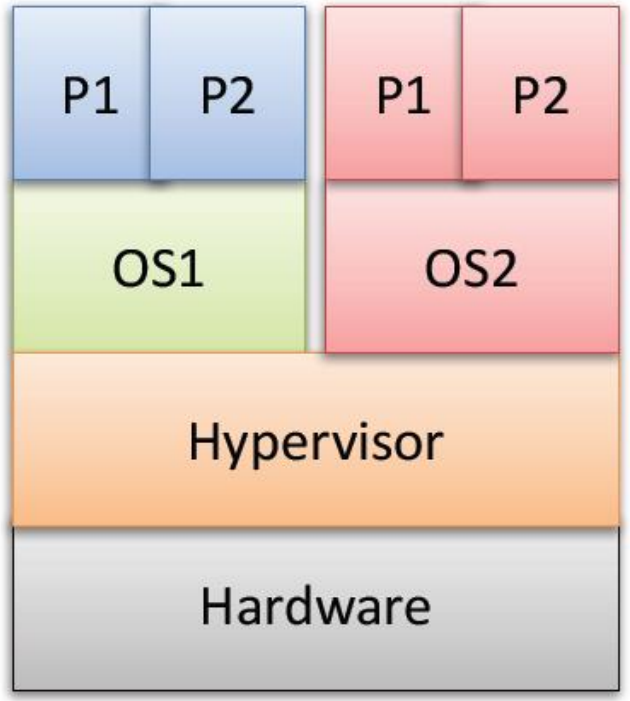
Virtualization use case – server consolidation

- Consolidation
  - Use VMs to optimize use of hardware
  - Pack as many VMs onto each server as possible
- Threat model?
  - Containment
  - Isolation
  - Assume guests OS are/can be compromised



Violating containment

- Escape-from-VM
  - Vulnerability in VMM or host OS
- Somewhat rare, but exist and very high value



VMware vulnerability allows users to escape virtual environment

By Joab Jackson Feb 28, 2008

A new vulnerability found in some VMware products allows users to escape their virtual environments and muck about in the host operating system, penetration testing software firm Core Security Technologies announced earlier this week.

This vulnerability (CVE Name: CVE-2008-0923) could poise significant risks to enterprise users who are deploying VMware software as a secured environment.

NEWS

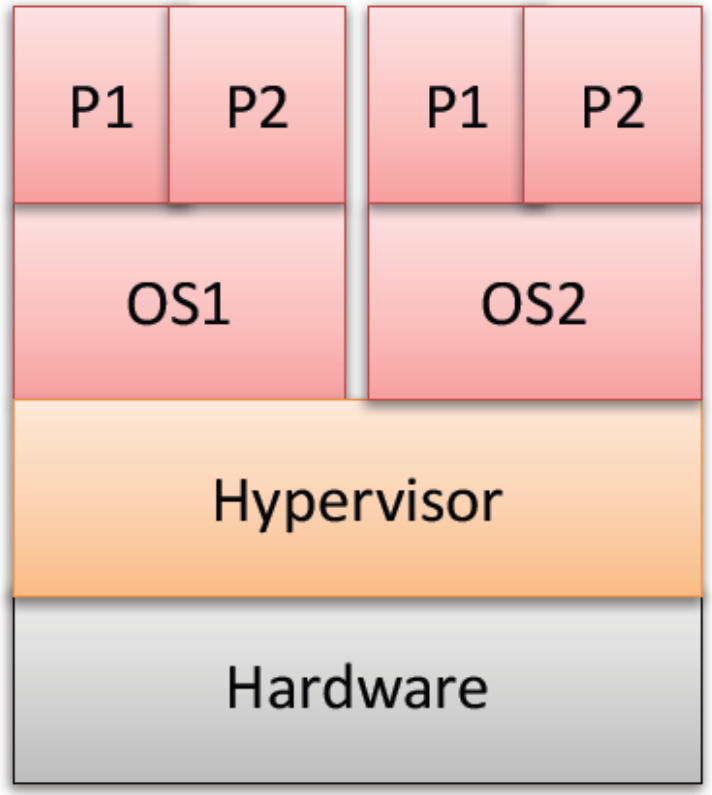
Xen hypervisor faces third highly critical VM escape bug in 10 months

The Xen paravirtualization mode is proving to be a constant source of serious vulnerabilities, allowing attackers to escape from virtual machines

https://blog.quarkslab.com/xen-exploitation-part-2-xsa-148-from-guest-to-host.html#id17

Violating isolation

- Covert channels between VMs circumvent access controls
  - Bugs in VMM
  - Side-effects of resource usage
- Degradation-of-Service attacks
  - Guests might maliciously contend for resources
  - Xen scheduler vulnerability
    - (https://arxiv.org/abs/1103.0759)
- Side channels
  - Spy on other guest via shared resources
  - Cross-VM cryptographic side-channel attacks [Zhang, Juels, Reiter, R. – CCS `12]



OS level virtualization

- Chroot jail
  - System call that virtualizes the file system. It basically changes the “root” folder for a process
- FreeBSD jails / Linux Vserver
  - System calls / kernel patch
  - Virtualizing file system
  - Resource limits (CPU, Memory)
  - Networking subsystem
  - No guest OS, one kernel for virtualized instances

Sandboxing

- Avoid modification user data or data of other applications
- Restricts capabilities for interacting with the operating system

- Android, IOS
  - Permissions for an App to access OS resources

Summary

- Mandatory access control (MAC) vs Discretionary access control (DAC)
  - Biba model, Bell-LaPadula, Domain and Type enforcement
- Separation of applications
  - Very strong implementation: virtualization
    - Multiple virtual machines running on the same physical hardware
    - Different ways to implement this
      - Full virtualization
      - HW-assisted virtualization
      - Paravirtualization
      - Host-based virtualization
  - Weaker way to compartmentalize: sandboxing
    - Sandboxing limits access to resources
  - Attacks typically aim at breaking out of the jail

Ad-hoc solutions for better security

- Completely re-designing an OS is expensive
- More feasible: Add-on security for existing OS
- Multiple techniques:
  - Memory protection (NX bit) and ASLR (last lecture)
  - Compartmentalization and virtualization
  - Add mandatory access control
- **Detect (or prevent) intrusions and malware**
  - IDS, IPS, AV, firewall

IDS, IPS, AV

- Two kinds of intrusion-detection systems (IDS):
  - Network-based IDS (NIDS)
  - Host-based IDS (HIDS)
  - Distributed or hybrid IDS
- Some systems have additional capabilities to prevent intrusion
- Those systems are called intrusion-prevention systems (IPS), again:
  - Network-based IPS (NIPS)
  - Host-based IPS (HIPS)
- Special kind of HIDS (HIPS): antivirus software (AV)
- AV is typically more generally anti-malware software (aka virus scanners, malware scanners)

Malware

Definition

- Malware is malicious software or functionality that a user does not intend to run.

- Viruses, Worms, Trojans
- Rootkits, Bootkits
- Botnets
- Ransomware, Adware, Spyware, Scareware, …

Malware detection

- Idea: look at incoming files before they are stored on the hard drive
- Scan for malware, stop if malware detected
- Important malware-scanner characteristics:
  - Detection rate: percentage of malware that is detected
  - Undetected malware is called false negatives
  - Files that are incorrectly classified as malware are false positives
- Mainly two techniques to detect malware:
  - Signature-based detection: (look for known patterns in files)
  - Heuristic detection: Analyse behavior and make decision

Signature-based detection

- Signature-based malware detection only detects known malware
  - Signatures can be as simple as a cryptographic hash
- Essential requirement: update the signature database daily
  - Still cannot detect zero-day (next-generation) malware
- Typically look for certain code sequences (less susceptible to minor changes)

- Code polymorphism can avoid signature detection
  - Use automated engine to generate many versions of a virus (All have the same functionality, but look different)
  - Use packers / encryption to “pack” the code and unpack the true malware in memory

Heuristic malware detection

- Run the malware in a safe environment (virtual machine, sandbox), study behavior
- Heuristic analysis relies on experience
- Good at detecting malware with behavior that “has been seen before”
- Typically not good at detecting really new malware
- Certainly not reliable at detecting new malware

AV can’t hurt, or can it?

- Common security recommendation for end users: “Use a malware scanner (AV) and keep it up to date”
- “Wisdom” behind this recommendation: AV certainly makes security better (even if it doesn’t detect everything)
- Multiple problems with this wisdom:
  - AV software can seriously degrade system performance
  - False positives can break system functionality
  - AV software is highly trusted (needs privileged access), but not necessarily trustworthy
  - Users may feel secure and behave less careful
  - AV software can actively degrade security (e.g. Kaspersky):
    - Kaspersky has man-in-the-middle functionality for SSL connections
    - Kaspersky spoke SSL 3.0 (although the browser had it disabled)
    - SSL 3.0 is vulnerable to the POODLE attack

Zip bombs

- Malware scanners (AV) needs to unpack zipped files
  - Unpacked copy needs to sit somewhere in memory or on disk
  - Can use this as attack against AV:
- Create small zip file, which expands to huge unpacked data
- Famous example: 42.zip
  - Packed size: 42 KB
  - Fully unpacked (after 5 levels of unzip): 4.5 PB
  - Expansion factor of >100,000,000,000
- Or create a zip file that contains itself
  - Virus scanners will keep unpacking forever
  - This exists, for details see http://research.swtch.com/zip
  - Not restricted to zip, also works with gzip

Host-based IDS (HIDS)

- HIDS goes beyond malware scanning (although there may be some overlap)
- Typically register certain resources with the IDS, those resources are monitored
  - system files, Windows registry entries, network ports
  - remember state of resource, detect modifications
- Additionally, analyze log files
  - Signature-based intrusion detection (similar to anti-virus)
  - Anomaly-based intrustion detection
    - hard to obtain good detection rate at low false-positive rate in highly dynamic systems

Anomaly-based intrustion detection

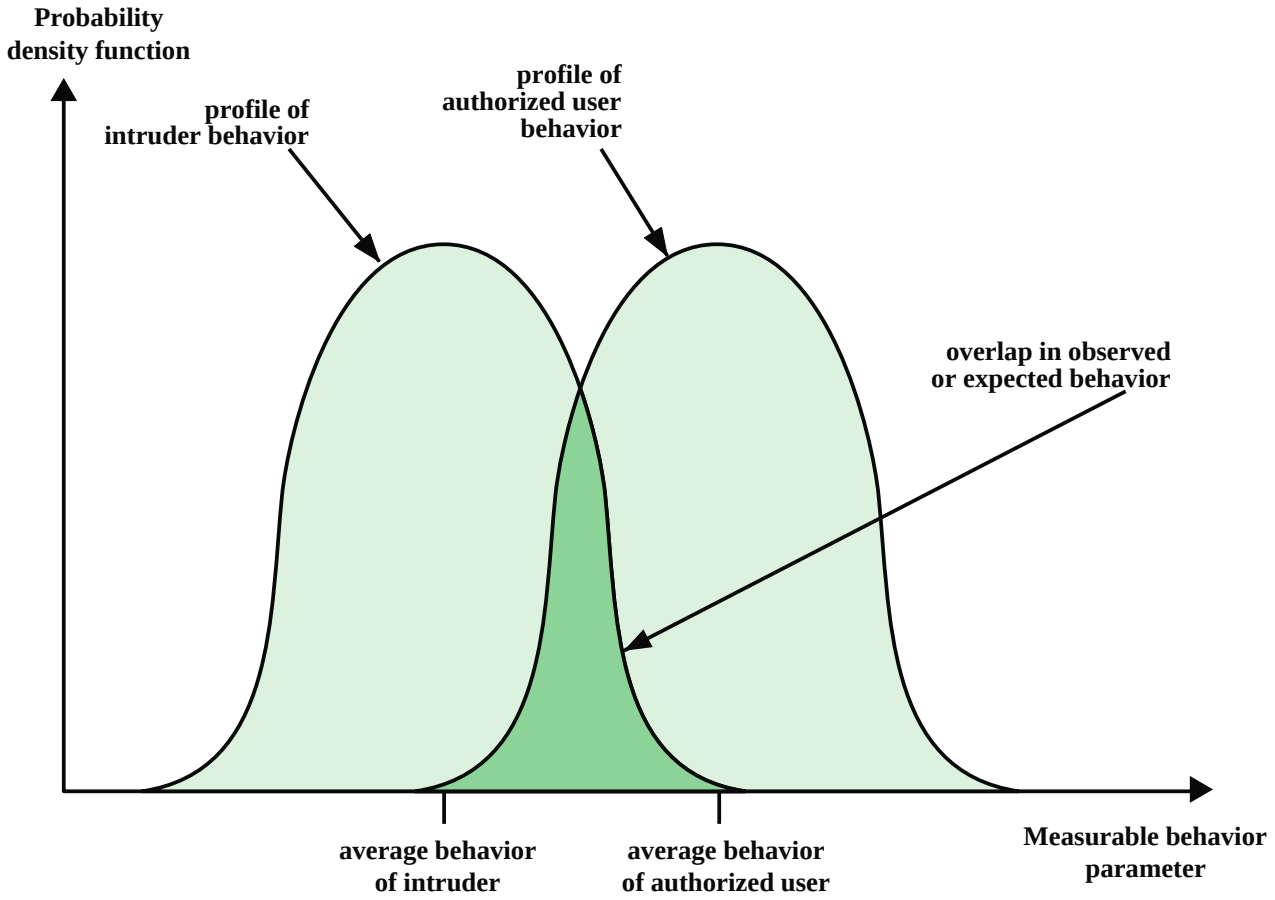


Figure 8.1 Profiles of Behavior of Intruders and Authorized Users

- Statistical
- Knowledge based
- Machine-learning

Intrusion Detection Techniques

Attacks suitable for

Signature detection

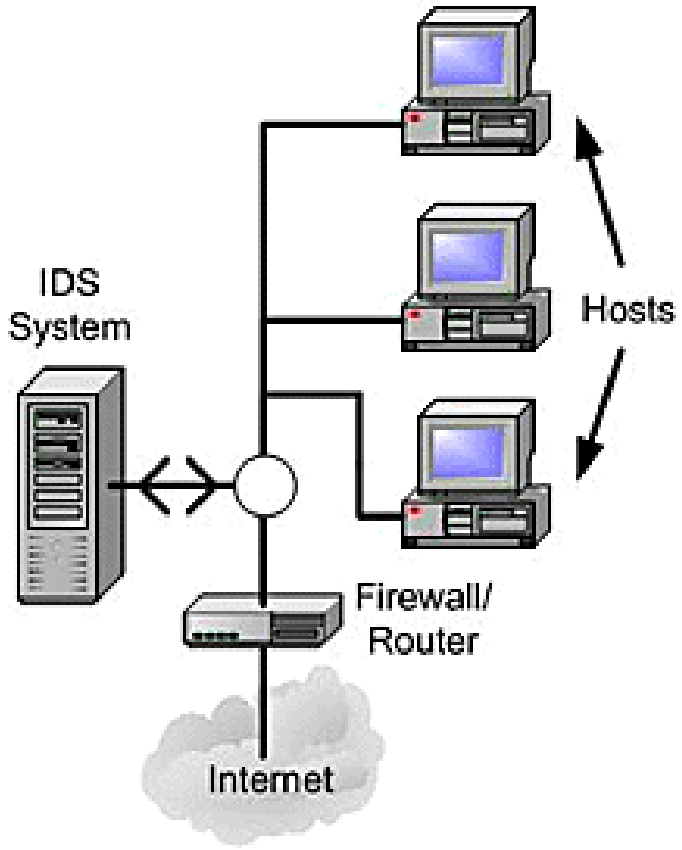
- Application layer reconnaissance and attacks
- Network / transport layer reconnaissance and attacks
- Unexpected application services
- Policy violations

Attacks suitable for

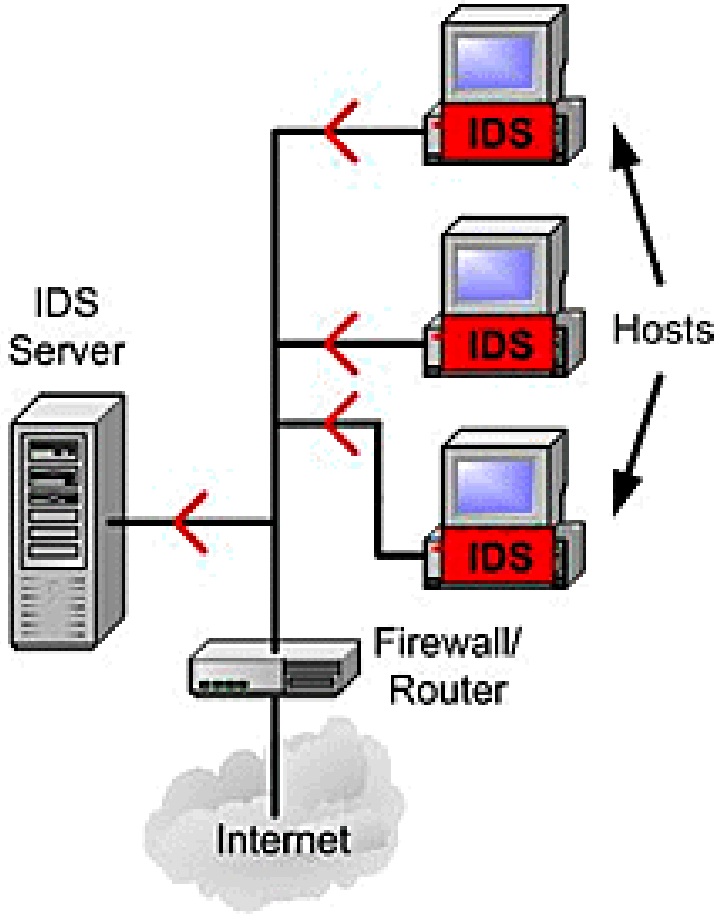
Anomaly detection

- Denial-of-service (DoS) attacks
- Scanning
- Worms

Network based IDS



Host based IDS

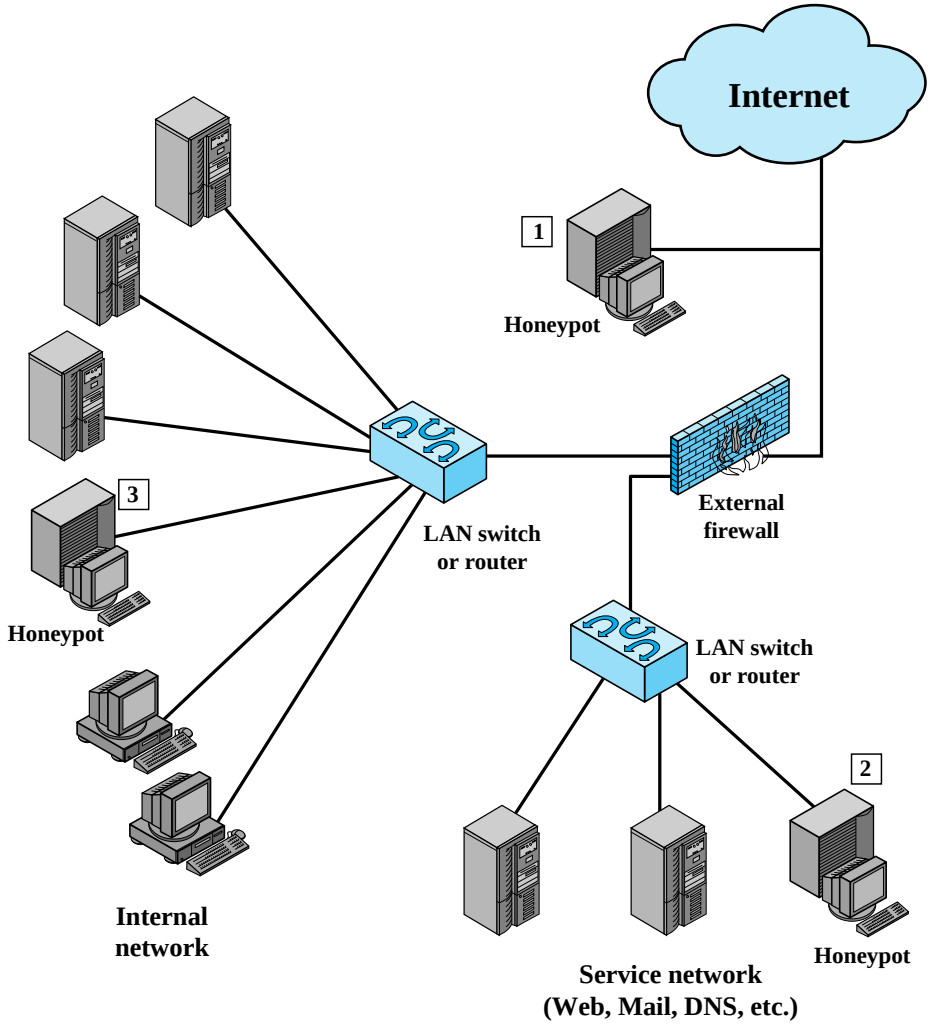


Network-Based IDS (NIDS)

- Monitors traffic at selected points on a network
- Examines traffic packet by packet in real or close to real time
- May examine network, transport, and/or application-level protocol activity
- Comprised of a number of sensors, one or more servers for NIDS management functions, and one or more management consoles for the human interface
- Analysis of traffic patterns may be done at the sensor, the management server or a combination of the two

Honeypots

- Decoy systems designed to:
  - Lure a potential attacker away from critical systems
  - Collect information about the attacker's activity
  - Encourage the attacker to stay on the system long enough for administrators to respond
- Systems are filled with fabricated information that a legitimate user of the system wouldn't access
- Resources that have no production value
  - Incoming communication is most likely a probe, scan, or attack
  - Initiated outbound communication suggests that the system has probably been compromised

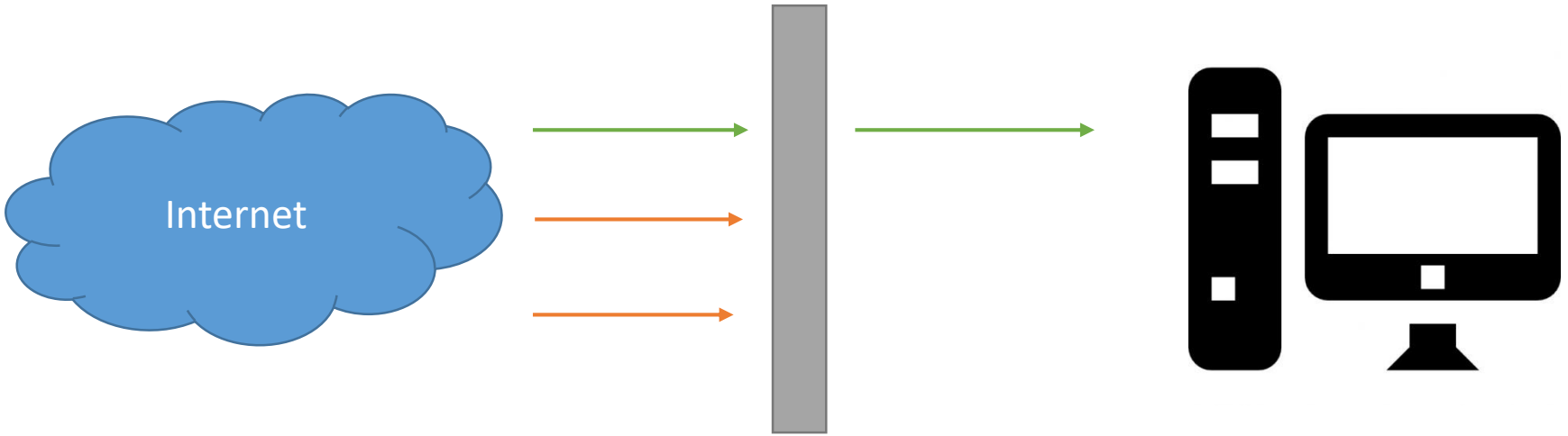


Recover after intrusion

- Easy situation:
  - download a file from the Internet, AV complains
    => Don’t run/open file, but stop download (or delete)
- Hard situation: AV complains about old files (or IDS reports intrusion)
  - AV software typically offers to “remove the virus/worm/trojan”
  - Question: Is that enough?
    - No.
  - Once a system has been compromised, you don’t know what else is broken
- Only reasonable recovery from intrusion:
  - Isolate the system (to prevent further damage)
  - Analyze what what compromised and how (forensics)
  - Restore to a clean state (reinstall, restore clean data backup)

Firewall

- Filters traffic by
  - Typically: IP address and protocol (limit access to a specific service)
  - Additionally:
    - Monitors and filter application protocol
    - Monitors network activity (time, rate of network requests)
    - Filters only requests from authenticated users



Denial of service

- “An action that prevents or impairs the authorized use of networks, systems, or applications by exhausting resources such as central processing units (CPU), memory, bandwidth, and disk space.”
- Categories of resources that could be attacked are
  - Network bandwidth
  - System resources
    - E.g. overload or crash network handling software
  - Application resources
    - E.g. exhaust application server requests limit

Classic DoS Attacks

- Flooding ping command
- Aim of this attack is to overwhelm the capacity of the network connection
- Traffic can be handled by higher capacity links on the path, but packets are discarded as capacity decreases
- Source of the attack is clearly identified unless a spoofed address is used
- Network performance is noticeably affected

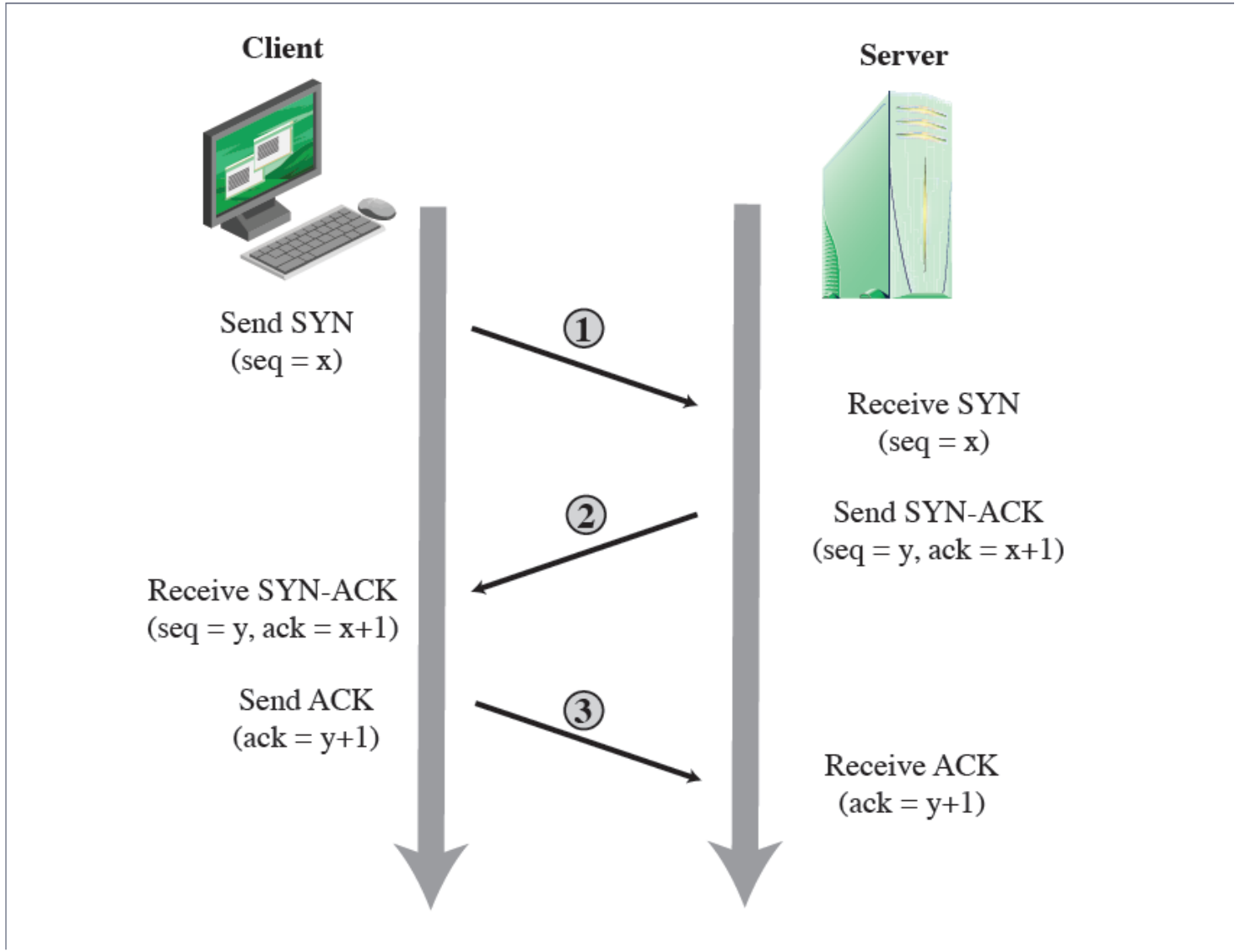
Source Address Spoofing

- Use forged source addresses
  - Usually via the raw socket interface on operating systems
  - Makes attacking systems harder to identify
- Attacker generates large volumes of packets that have the target system as the destination address
- Congestion would result in the router connected to the final, lower capacity link
- Requires network engineers to specifically query flow information from their routers

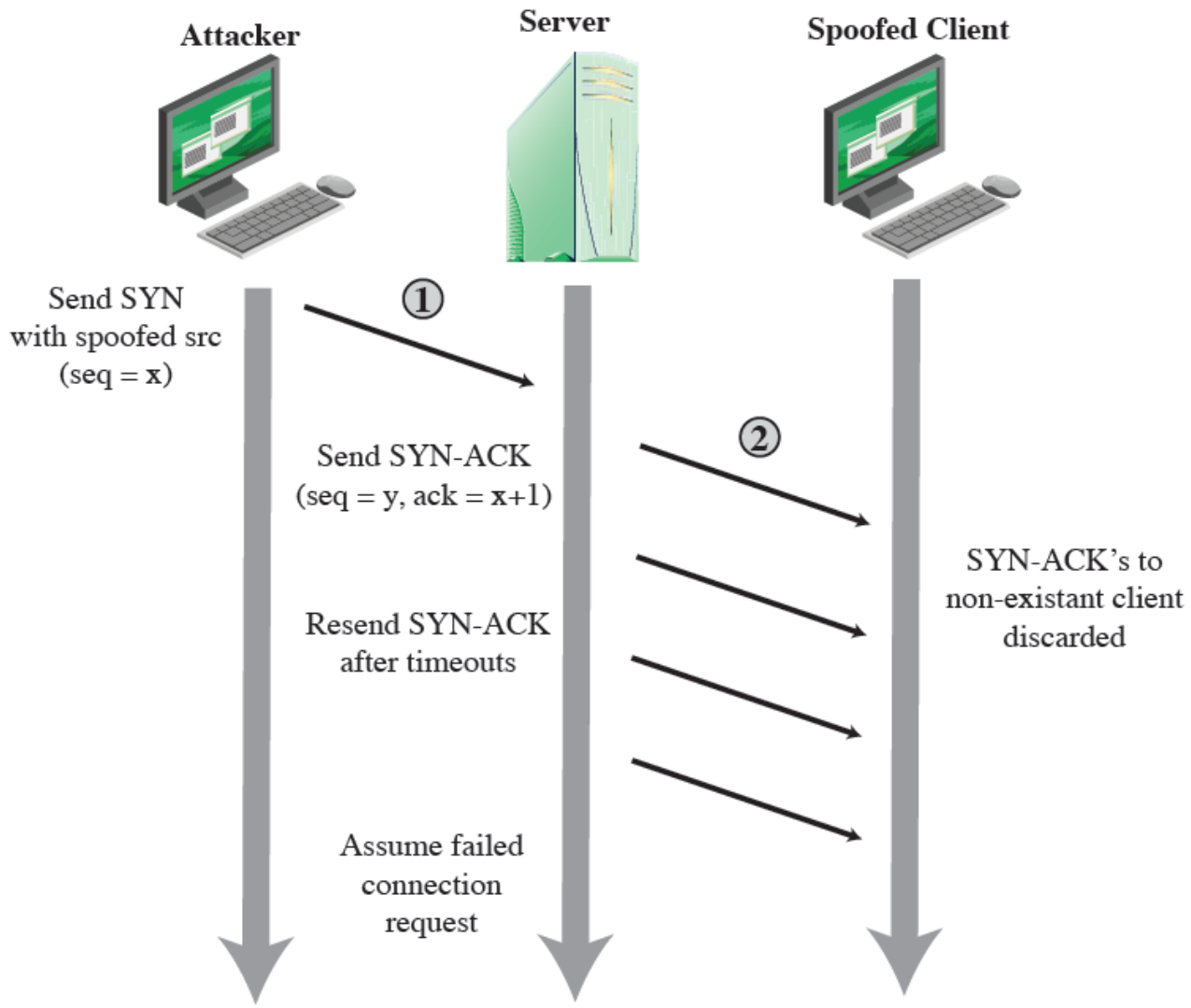
SYN Spoofing

- Common DoS attack
- Attacks the ability of a server to respond to future connection requests by overflowing the tables used to manage them
- Thus legitimate users are denied access to the server
- Hence an attack on system resources, specifically the network handling code in the operating system

- TCP three-way connection handshake:

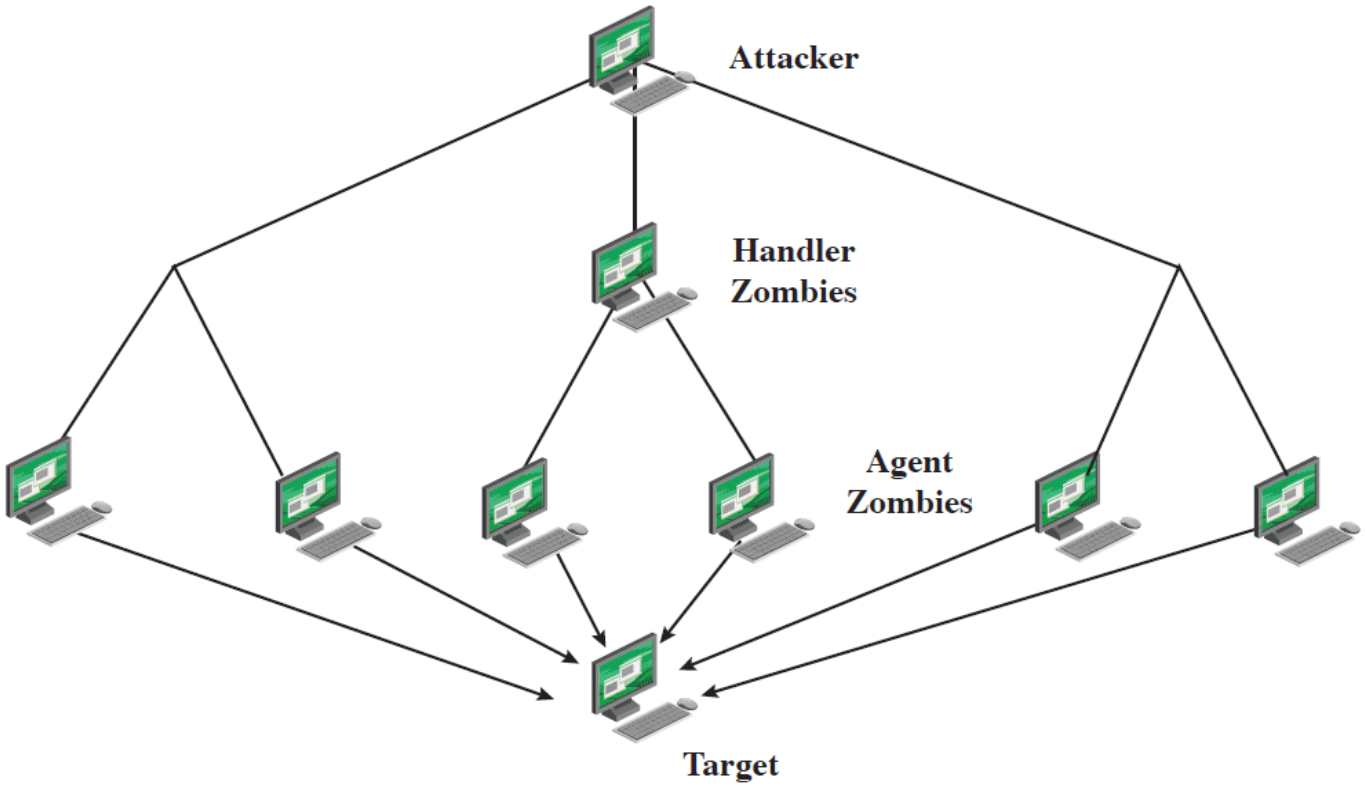


- TCP SYN spoofing attack:



Distributed Denial of Service (DDoS) Attacks

- Use botnets to send requests from many IPs
- Or utilize flaw in a widely used application
  - E.g. wordpress



HTTP Based Attacks

- Attempts to monopolize by sending HTTP requests that never complete
- Eventually consumes Web server’s connection capacity
- Utilizes legitimate HTTP traffic
- Existing intrusion detection and prevention solutions that rely on signatures to detect attacks will generally not recognize this

DoS Attack Defenses

- These attacks cannot be prevented entirely
- High traffic volumes may be legitimate

- Use mirrored and replicated servers
- Large datacenters can mitigate DDoS attacks

- Block spoofed source addresses
  - On routers as close to source as possible
- Apply antispoofing, directed broadcast, and rate limiting filters
- Use network monitors and IDS to detect and notify abnormal traffic paFerns

Penetračné testovanie

- „ ... attack on a computer system with the intention of finding security weaknesses, potentially gaining access to it, its functionality and data.“ – Wiki
- 2 základné typy pentestu
  - „white box“ – určité info sú známe, napríklad výstup vulnerability assesmentu
  - „black box“ – simulácia útoku bez znalosti prostredia (zero-knowledge)
- Podľa predmetu testovania:
  - interný– LAN
  - externý– webové aplikácie a služby
- Metóda testovania:
  - automatické – scanovacie softvéry, online nástroje, skripty, ...
  - manuálne – najmä na overenie výsledkov, vylúčenie FP

Postup testovania

Čo by robil útočník?

- Reconnaisance
  - Získavanie informácií o cieli: IP adresy, DNS info, typ/verzia systému, dáta o zamestnancoch, ich kontách a prihlasovacích údajoch
- **Scanning and Enumeration**
  - Súhrn znalostí o cieli, identifikácia, ktoré časti môžu byť zraniteľné
- **Penetration**
  - Získanie prístupu zneužitím zraniteľností a obídením bezpečnostných mechanizmov
- Denial of Service
- Escalation and Maintaining Access
  - Kroky na vytvorenie a udržanie stabilného a nenápadného prístupu do systému - napríklad inštaláciou rootkitu.
- Covering Tracs and Hiding

Postup testovania

- Stanovenie okruhu testovania (scope)
- Získavanie informácií o cieli
- Pokusy o exploitáciu na získanie prístupu do systému a eskaláciu privilégií
  - Scanery, manuálne testovanie a útoky
  - Napr. testovanie najčastejších zraniteľností podľa OWASP TOP 10
- Pokus o získanie citlivých dát
- Upratovanie – zahladenie stôp, a reportovanie výsledkov

Penetračné testovanie - nástroje

- Komerčné scannery a toolkity
  - Acunetix WVS
  - Burpsuite Pro
    - Scanner, Spider, Intruder, Repeater, Decoder, ...
  - Nessus
- Open-source
  - Skipfish
  - cURL
  - SQLMap
  - Nmap/Zenmap
  - Doplnky prehliadačov – TamperData, WebScarab

Záver

Typické bezpečnostné problémy

- chyby v softvéri
  - nedostatočná kontrola vstupov
    - buffer overflow
    - špeciálne znaky
- race conditions
  - časové okno medzi kontrolou a vykonaním operácie
- používanie nebezpečných funkcií
  - strcpy, gets, scanf
- nesprávne používanie funkcií
  - printf (vstup ako formátovací reťazec)

Typické bezpečnostné problémy

- príliš veľké práva programov
  - root (UNIX), system (Windows)
- zneužívanie chýb v programoch na spustenie kódu
  - web browser, e-mail klient, ...
  - bez MAC plný prístup k objektom používateľa
- dôveryhodnosť administrátora
  - neobmedzené práva
- nezodpovední používatelia
  - každý používateľ je zodpovedný za bezpečnosť
  - používateľ nechráni len seba, ale aj systém ako celok

Typické bezpečnostné problémy

- nedostatočná informovanosť o bezpečnosti
  - používatelia
  - administrátori
  - vývojári

Pozor na čiastočné riešenia

- riadenie prístupu
  - prístup priamo k zariadeniu
- šifrovanie, elektronické podpisovanie
  - „odchytenie“ kľúčov, hesiel
- overovanie elektronického podpisu
  - podvrh dôveryhodných verejných kľúčov
- reziduálna informácia
  - disky
  - RAM
- nezabezpečené prístupy k HW (USB, PCIe, …)