Security in Operating System

Tej Bahadur Shahi
Asst. Prof.
CDCSIT,TU

System Protection (Preview)

- Protecting files and other resources from accidental misuse by cooperating users sharing a system.
- This is for normal purposes

System Security

- Deals with protecting systems from deliberate attacks, either internal or external individuals.
- Individuals intentionally attempts to steal information, damage information.

Types of Violations

- Basically there are following breach of security:
  - ***Breach of Confidentiality***
  - ***Breach of Integrity***
  - ***Breach of Availability***

This is a commonly known as CIA triad in information security Theory

Breach of Confidentiality

- Theft of private or confidential information,
- For example: credit-card numbers, trade secrets, patents, secret formulas, manufacturing procedures, medical information, financial information, etc.

Breach of Integrity

- **–**Unauthorized m***odification*** of data, which may have serious indirect consequences.
- For example a popular game or other program's source code could be modified to open up security holes on users systems before being released to the public.

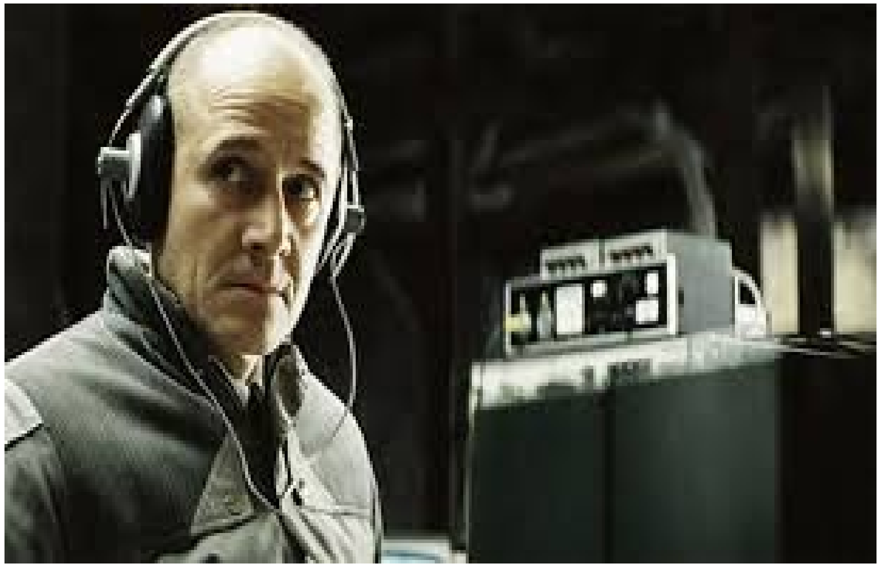
Breach of Availability

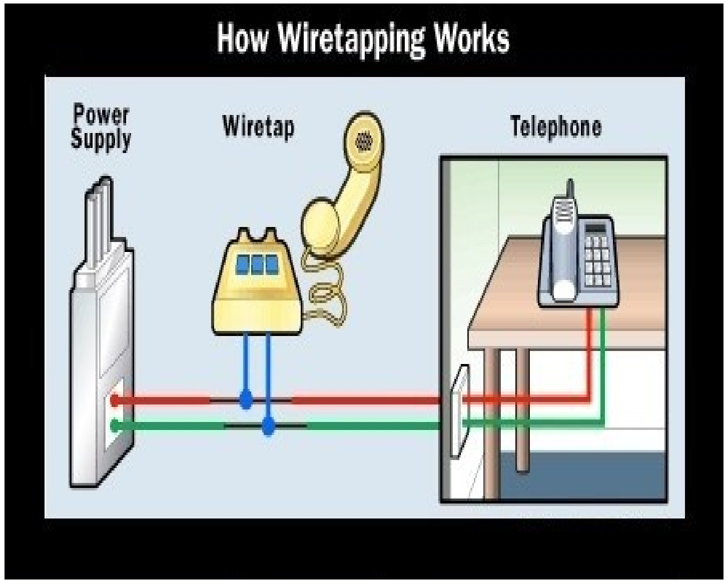
- **-**Unauthorized ***destruction*** of data, often just for the "fun" of causing havoc and for bragging rites. Vandalism of web sites is a common form of this violation.

Types of attack

- In general, there are following possible attack in computer system.
  - Active attack Vs Passive Attack
  - Inside Attack Vs Outside Attack
  - Passive Attack
    - Eavesdropping
    - Wiretapping/port scan etc
  - Active Attack
    - Denial of Service
    - Man in the middle attack

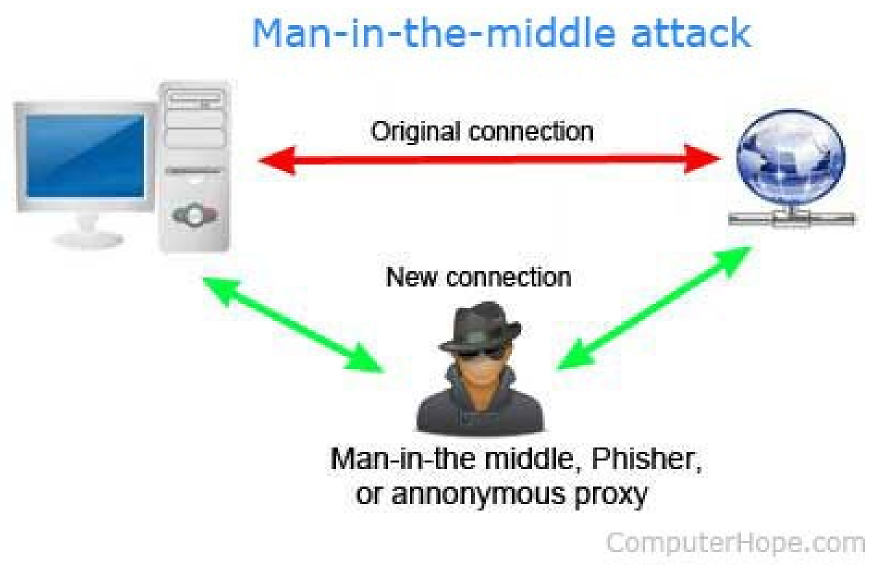
Some Pictures

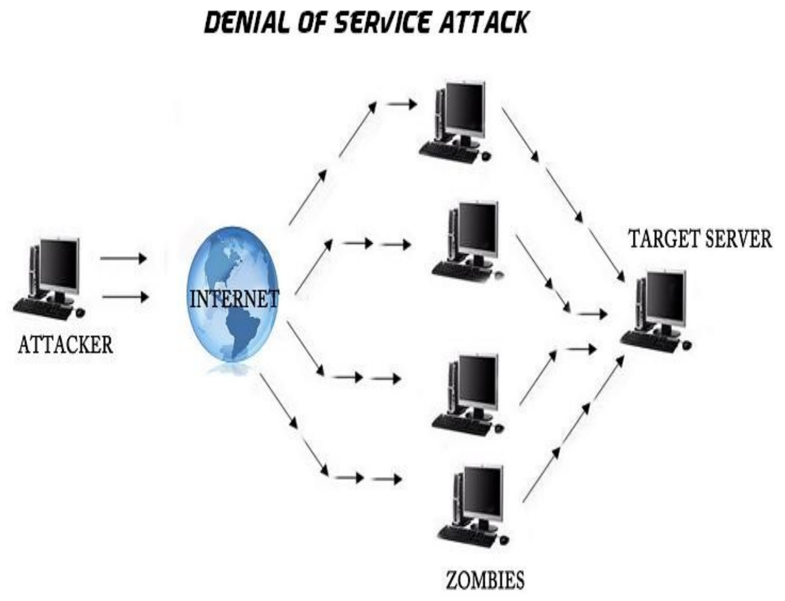




Eavesdropping and wiretapping

Some pictures





Picture Source: Google Image

Level of Protections

- There are four levels at which a system must be protected:
  - Physical
  - Human: ***social engineering,***
    - **Phishing**
    - **Dumpster Diving**
    - **Password Cracking**
  - **Operating System**
  - **Network**

Threat

- Program Threat
- System threat

Program Threat

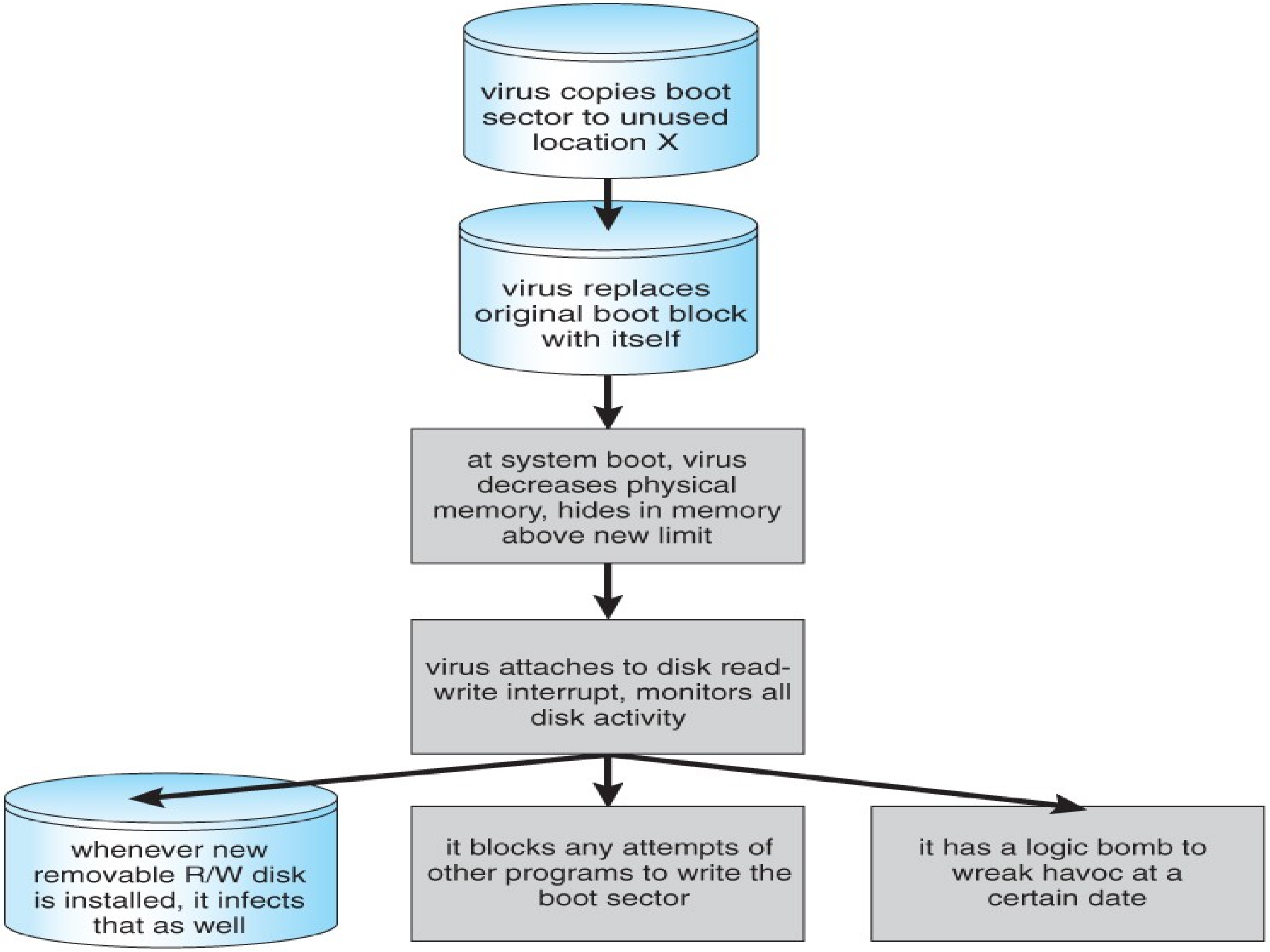
- A ***Trojan Horse*** is a program that secretly performs some maliciousness in addition to its visible actions.
- A classic Trojan Horse is a login emulator,
  - which records a users account name and password, issues a "password incorrect" message, and then logs off the system
  - The user then tries again (with a proper login prompt), logs in successfully, and doesn't realize that their information has been stolen

Threat: Trap Door

- A ***Trap Door*** is when a designer or a programmer (or hacker) deliberately inserts a security hole that they can use later to access the system
- A clever trap door could be inserted into a compiler, so that any programs compiled with that compiler would contain a security hole

Other threats

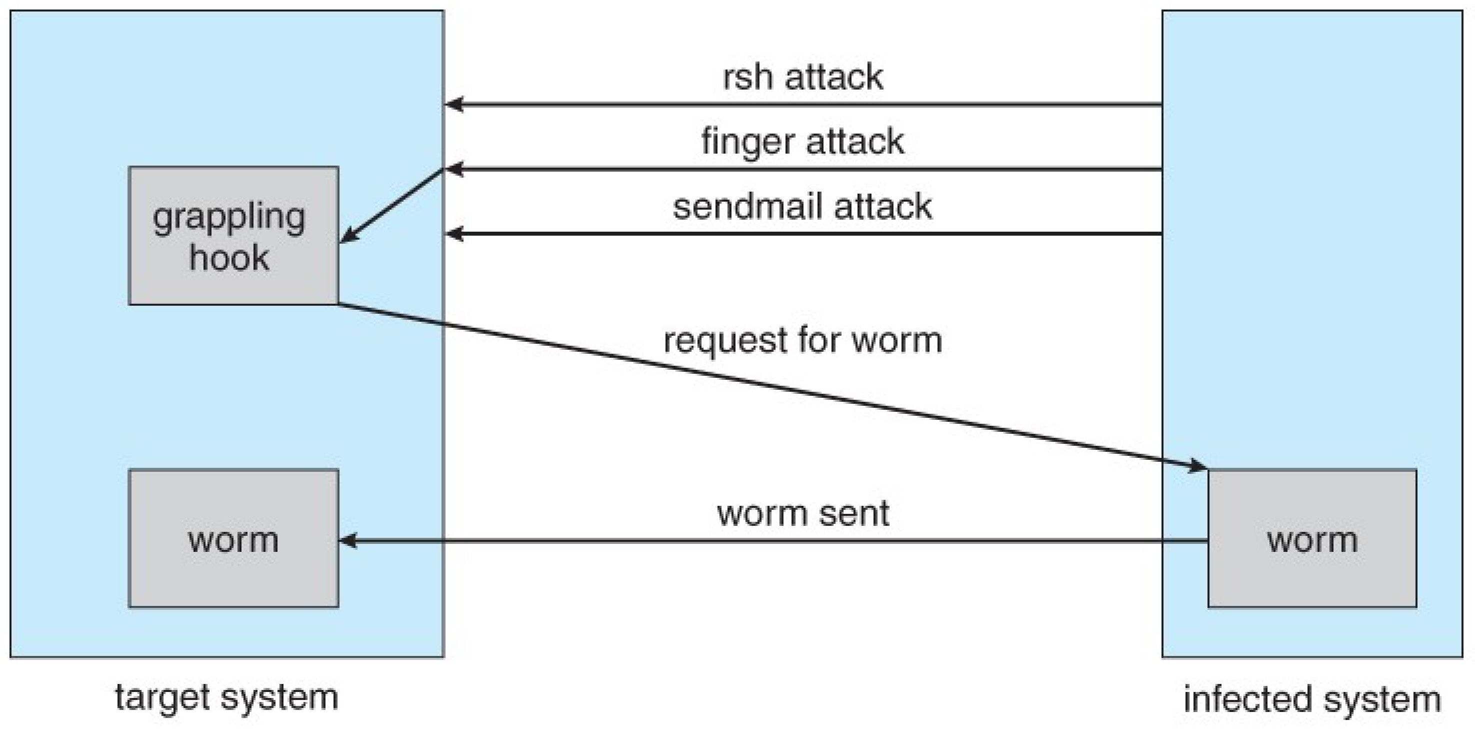
- **Logic Bomb**:
- **Stack and Buffer Overflow**
- **Viruses: d**esigned to replicate itself


virus copies boot sector to unused location X
virus replaces original boot block with itself
at system boot, virus decreases physical memory, hides in memory above new limit
virus attaches to disk read-write interrupt, monitors all disk activity
whenever new removable R/W disk is installed, it infects that as well
it blocks any attempts of other programs to write the boot sector
it has a logic bomb to wreak havoc at a certain date

System and Network Threats

- A ***worm*** is a process that uses the fork / spawn process to make copies of itself in order to wreak havoc on a system.
- Worms consume system resources, often blocking out other, legitimate processes.
- Worms that propagate over networks can be especially problematic, as they can tie up vast amounts of network resources and bring down large-scale systems.

The Morris Internet worm



Port Scanning

- ***Port Scanning*** is technically not an attack, but rather a search for vulnerabilities to attack.
- Try to connect to every node and attempt to contact.
- Once the port is determined, next step is to determine what daemon is listening, and is there any flaw in the version of daemon.
- ***nmap*** (http://www.insecure.org/nmap) and ***nessus*** (http://www.nessus.org)

Denial of Service

- ***Denial of Service (DOS)*** attacks do not attempt to actually access or damage systems, but merely to clog them up so badly that they cannot be used for any useful work. Tight loops that repeatedly request system services are an obvious form of this attack.
- DDOS: distributed DOS

Cryptography as Security tools

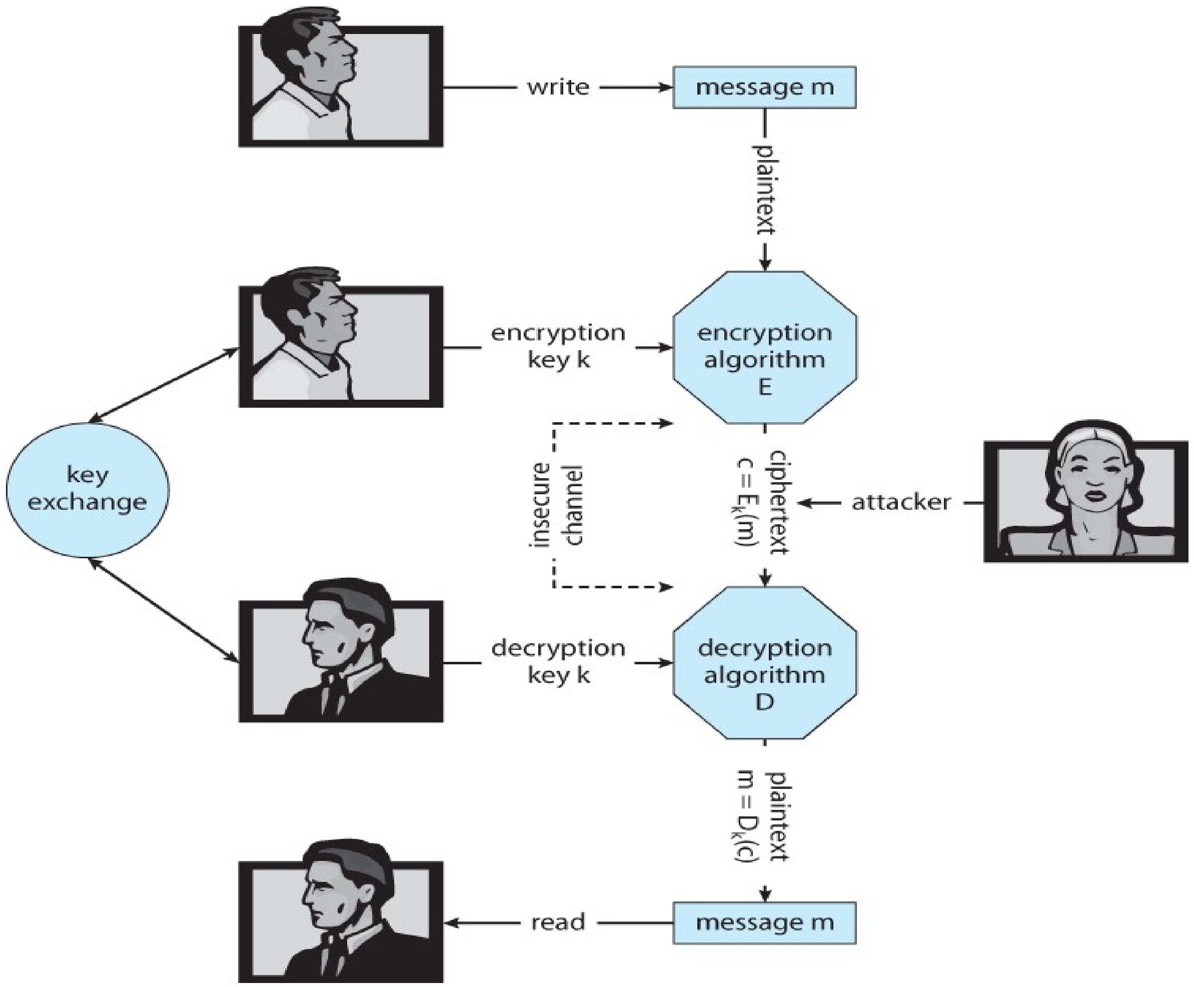
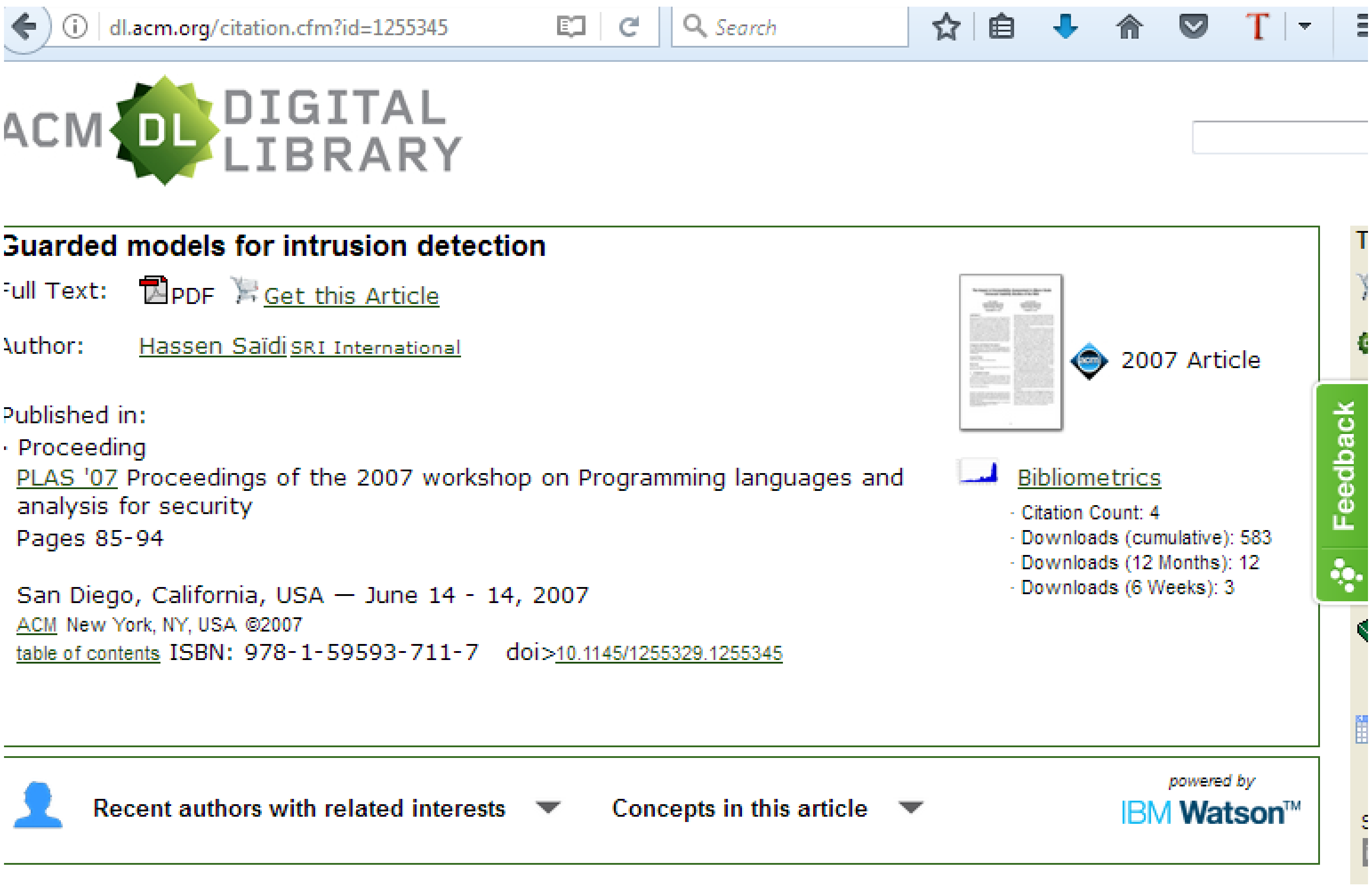


Fig: Encryption and Decryption, Source: Textbook

Implementation of Cryptography

- Network communications are implemented in multiple layers - Physical, Data Link, Network, Transport, and Application being the most common breakdown.
- At the network layer the most common standard is **IPSec,** a secure form of the IP layer, which is used to set up **Virtual Private Networks, VPNs.** At the transport layer the most common implementation is SSL.

Research Paper Review



dl.acm.org/citation.cfm?id=1255345

Guarded models for intrusion detection

Full Text: PDF Get this Article

Author: Hassen Saïdi SRI International

Published in:

· Proceeding
PLAS '07 Proceedings of the 2007 workshop on Programming languages and analysis for security
Pages 85-94

San Diego, California, USA — June 14 - 14, 2007
ACM New York, NY, USA ©2007
table of contents ISBN: 978-1-59593-711-7 doi>10.1145/1255329.1255345

2007 Article

Bibliometrics
· Citation Count: 4
· Downloads (cumulative): 583
· Downloads (12 Months): 12
· Downloads (6 Weeks): 3

Recent authors with related interests

Concepts in this article

powered by IBM Watson™

Guarded Models For Intrusion Detection

Hassen Saïdi
Computer Science Laboratory
SRI International
saidi@csl.sri.com

Abstract

Host-based intrusion detection systems that monitor an application execution and report any deviation from its statically built model have seen tremendous progress in recent years. However, the weakness of these systems is that they often rely on overly abstracted models that reflect only the control flow structure of programs, and therefore are subject to so-called "mimicry attacks". Authors of these models have argued that capturing more of the data flow characteristics of a program is necessary to prevent a large class of attacks, in particular, non-control-data attacks. In this paper, we present the guarded model, a novel model that addresses the various deficiencies of the state-of-the-art intrusion detection systems. Our model is a generalization of previous models that offers no false alarms, a very low monitoring overhead, and is automatically generated. Our model detects mimicry attacks by combining control flow and data flow analysis, but can also tackle the ever increasingly threatening non-control-data flow attacks. Our model is the first model built automatically by combining control flow and data flow analysis using state-of-the-art tools for automatic generation and propagation of invariants. Our model not only prevents intrusions, but allows in some cases the detection of application logic bugs. Such bugs are beyond the reach of current intrusion detection systems.

Categories and Subject Descriptors D [4]: 6 Security and Protection

General Terms Security, Verification

Keywords Static analysis, dynamic analysis, invariant generation, intrusion detection

program. Detection of attacks rely on the fact that code injection often results in the execution of a system call that is not invoked by the application at all, or a sequence of system calls that are executed in an order not allowed by the application code. Finally, false alarms are eliminated because the model is an overapproximation of the behavior of the application. Therefore, while the model might be porous and not detect attacks, it will never raise a false alarm. Starting with Wagner and Dean's paper, various models capturing the sequences of legitimate system calls have been proposed [32, 18, 13, 16, 8].

Current state-of-the-art host-based intrusion detection systems approaches struggle with several issues. The most important one is the precision of the model. The more precise the model is, the less an attack is possible. Current models are limited when it comes to handling complex control structures such as loops, recursive function calls, and circuits in function calls. These limitations produce today's models in forms of very crude overapproximations of the actual behavior of the application and are all subject to so-called *mimicry attacks* [33] that produce a sequence of system calls that conforms to the model but is not a sequence in the original code. The imprecision of state-of-the-art models comes from the weakness of the static analysis methods employed to construct them. These weaknesses can be summarized by the absence of any significant data flow analysis. Mimicry attacks exploit the nondeterminism that arises from the imprecision of the models. A new wave of attacks called, *non-control-data attacks* [10] exploit the nondeterminism in the application code itself and therefore allow the subversion of the intended data flow. We observe that the attacks mentioned above are all preventable by making sure that the application models used for intrusion detection capture as precisely as possible

Thank You