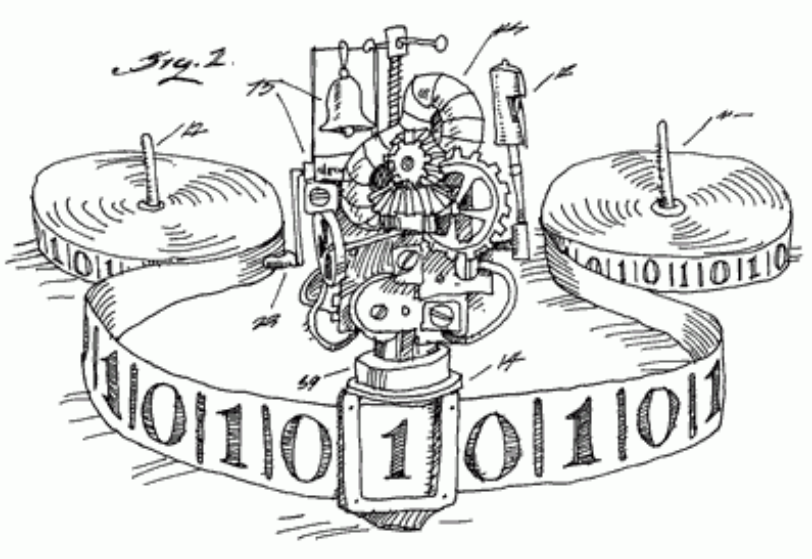
INFO 101 – Introduction to Computing and Security

[2020 - Week 8 / 1]



Prof. Dr. Rui Abreu

University of Porto, Portugal

rui@computer.org

@rmaranhao

Classes & Final Exam

- December 7 – 11 (next week) is our last week with classes;
- There will be an exam in January
  - Prof. Haitao Zhao will send you more details soon!
  - The exam consists of two parts:
    - Part 1: 25 multiple choice questions
    - Part 2: 4 exercises
  - Duration: 90 minutes

What we are going to discuss…

- Introduction to security, ethics, and privacy
- Discussing the importance of security for information systems
- Basic foundations behind computer security including vulnerabilities, attacks, and their consequences
- Discuss the need for verification and validation to avoid bugs and vulnerabilities

Recent security case

Marriott data breach / Starwood hacked (Nov. 30th 2018)

Marriott International has been investigating a hack involving unauthorized access to the guest reservation database at its Starwood unit since 2014, in what may be one of the biggest such data breaches.

The attack is troubling not just because of its sheer size, but also the level of detail potentially stolen by the attackers. The hack affects some 500 million guests, and for about 327 million of them, the data included passport numbers, emails and mailing addresses, Marriott said.

Marriott International has been investigating a hack involving unauthorized access to the guest reservation database at its Starwood unit since 2014, in what may be one of the biggest such data breaches.

The attack is troubling not just because of its sheer size, but also the level of detail potentially stolen by the attackers. The hack affects some 500 million guests, and for about 327 million of them, the data included passport numbers, emails and mailing addresses, Marriott said.

Text from MSN / Bloomberg - http://a.msn.com/00/en-us/BBQilJ2?ocid=se

The Marriott hack may rank only below Yahoo as one of the biggest of personal data, when 3 billion users were exposed to a 2013 security breach. Marriott shares slumped 5.6 percent in pre-market trading.

Regulators and consumers have been stepping up their action against companies that have suffered security breaches as such attacks have increasingly become more severe. Target Corp. last year agreed to pay $18.5 million to settle investigations by dozens of states over a 2013 hack of its database in which the personal information of millions of customers was stolen, while Equifax is facing billion-dollar law suits and a regulatory investigation.

Securities in Madrid. "This is yet another company that has been hit by a hacking and a reminder to any company that manages customers's personal data that they need to work harder to protect them from future attacks."

Marriott's statement indicates the hacking was going on years before the company acquired Starwood in a deal valued at about $13.6 billion that closed in September 2016. Marriott's database contained guest information relating to reservations at Starwood properties on or before Sept. 10, 2018. For some, it also included payment card details, said Marriott, which didn't identify who the perpetrators might be.

Security – a brief introduction

security

noun · **UK** /sɪˈkjʊə.rə.ti/ **US** /səˈkjʊr.ə.ţi/

security *noun* **(PROTECTION)**

B1 [U] **protection of a person, building, organization, or country against threats such as crime or attacks by foreign countries:**

[U, + sing/pl verb] **the group of people responsible for protecting a building:**

[Cambridge Dictionary]

There is always a fight going on..

- Between the **attackers** and **defenders** (of our systems)!!!!

Let us start with a simple example

A simple example

- Given:
  - Web page providing the user name as argument
  - Server side script

HTML page content (Client):

```
<form action="test.cgi" method=GET>
<input maxlength=10 type="input"
          name="ARGV">Username</input>
</form>
```

Script `test.cgi` (Server):

```
$username = $ARGV;
system("cat /logs/$username" . ".log");
```

A simple example

HTML page sends content:

```
http://to_server/test.cgi?ARGV=Mary
```

Script `test.cgi` (Server):

```
$username = $ARGV;
system(“cat /logs/$username“ . “.log“);
```

HTML page may send also:

```
http://to_server/test.cgi?username=TOO_LONG_
FOR_A_USERNAME
```

We might also use this to submit other parameters like:

```
../etc/passwd
```

```
$username = "../etc/passwd";
system("cat /logs/$username" . ".log");
```

or

```
Mary.log; rm –rf /; cat blah
```



```
$username = "Mary.log; rm –rf /; cat blah";
system("cat /logs/$username" . ".log");
```

Avoiding trouble!?

Have a look at attack patterns

- See for example: `http://capec.mitre.org/`

- Or: `https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/attack.html`

OWASP

- Open Web Application Security Project
- Top 10 (2020):
  - A1 Injection
  - A2 Broken Authentication and Session Management
  - A3 Cross-Site Scripting (XSS)
  - A4 Insecure Direct Object References
  - A5 Security Misconfiguration
  - A6 Sensitive Data Exposure
  - A7 Missing Function Level Access Control
  - A8 Cross-Site Request Forgery (CSRF)
  - A9 Using Components with Known Vulnerabilities
  - A10 Unvalidated Redirects and Forwards

OWASP A1 - Injection

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends simple text-based attacks that exploit the syntax of the targeted interpreter. Almost any source of data can be an injection vector, including internal sources.	Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or NoSQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc. Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws.		Injection can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover.	Consider the business value of the affected data and the platform running the interpreter. All data could be stolen, modified, or deleted. Could your reputation be harmed?

Am I Vulnerable To 'Injection'?

The best way to find out if an application is vulnerable to injection is to verify that all use of interpreters clearly separates untrusted data from the command or query. For SQL calls, this means using bind variables in all prepared statements and stored procedures, and avoiding dynamic queries.

Checking the code is a fast and accurate way to see if the application uses interpreters safely. Code analysis tools can help a security analyst find the use of interpreters and trace the data flow through the application. Penetration testers can validate these issues by crafting exploits that confirm the vulnerability.

Automated dynamic scanning which exercises the application may provide insight into whether some exploitable injection flaws exist. Scanners cannot always reach interpreters and have difficulty detecting whether an attack was successful. Poor error handling makes injection flaws easier to discover

How Do I Prevent 'Injection'?

Preventing injection requires keeping untrusted data separate from commands and queries.

1. The preferred option is to use a safe API which avoids the use of the interpreter entirely or provides a parameterized interface. Be careful with APIs, such as stored procedures, that are parameterized, but can still introduce injection under the hood.
2. If a parameterized API is not available, you should carefully escape special characters using the specific escape syntax for that interpreter. OWASP's ESAPI provides many of these escaping routines.
3. Positive or "white list" input validation is also recommended, but is not a complete defense as many applications require special characters in their input. If special characters are required, only approaches 1. and 2. above will make their use safe. OWASP's ESAPI has an extensible library of white list input validation routines.

Example Attack Scenarios

Scenario #1: The application uses untrusted data in the construction of the following **vulnerable** SQL call:

```
String query = "SELECT * FROM accounts WHERE custID='" +
request.getParameter("id") + "'";
```

Scenario #2: Similarly, an application's blind trust in frameworks may result in queries that are still vulnerable, (e.g., Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" +
request.getParameter("id") + "'");
```

In both cases, the attacker modifies the 'id' parameter value in her browser to send: ' or '1'='1. For example:

```
http://example.com/app/accountView?id=' or '1'='1
```

This changes the meaning of both queries to return all the records from the accounts table. More dangerous attacks could modify data or even invoke stored procedures.

References

OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Query Parameterization Cheat Sheet
- OWASP Command Injection Article
- OWASP XML eXternal Entity (XXE) Reference Article
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing

External

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection

Threat Agents	Attack Vectors	Security Weakness	
Application Specific	**Exploitability** **EASY**	**Prevalence** **COMMON**	**Detectability** **AVERAGE**
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends simple text-based attacks that exploit the syntax of the targeted interpreter. Almost any source of data can be an injection vector, including internal sources.	Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or NoSQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc. Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws.	

Technical Impacts	Business Impacts
Impact **SEVERE**	**Application / Business Specific**
Injection can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover.	Consider the business value of the affected data and the platform running the interpreter. All data could be stolen, modified, or deleted. Could your reputation be harmed?

Am I Vulnerable To 'Injection'?

The best way to find out if an application is vulnerable to injection is to verify that all use of interpreters clearly separates untrusted data from the command or query. For SQL calls, this means using bind variables in all prepared statements and stored procedures, and avoiding dynamic queries.

Checking the code is a fast and accurate way to see if the application uses interpreters safely. Code analysis tools can help a security analyst find the use of interpreters and trace the data flow through the application. Penetration testers can validate these issues by crafting exploits that confirm the vulnerability.

Automated dynamic scanning which exercises the application may provide insight into whether some exploitable injection flaws exist. Scanners cannot always reach interpreters and have difficulty detecting whether an attack was successful. Poor error handling makes injection flaws easier to discover

Example Attack Scenarios

Scenario #1: The application uses untrusted data in the construction of the following **vulnerable** SQL call:

```
String query = "SELECT * FROM accounts WHERE custID='" +
request.getParameter("id") + "'";
```

Scenario #2: Similarly, an application's blind trust in frameworks may result in queries that are still vulnerable, (e.g., Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" +
request.getParameter("id") + "'");
```

In both cases, the attacker modifies the 'id' parameter value in her browser to send: ' or '1'='1. For example:

```
http://example.com/app/accountView?id=' or '1'='1
```

This changes the meaning of both queries to return all the records from the accounts table. More dangerous attacks could modify data or even invoke stored procedures.

How Do I Prevent 'Injection'?

Preventing injection requires keeping untrusted data separate from commands and queries.

1. The preferred option is to use a safe API which avoids the use of the interpreter entirely or provides a parameterized interface. Be careful with APIs, such as stored procedures, that are parameterized, but can still introduce injection under the hood.
2. If a parameterized API is not available, you should carefully escape special characters using the specific escape syntax for that interpreter. OWASP's ESAPI provides many of these escaping routines.
3. Positive or "white list" input validation is also recommended, but is not a complete defense as many applications require special characters in their input. If special characters are required, only approaches 1. and 2. above will make their use safe. OWASP's ESAPI has an extensible library of white list input validation routines.

References

OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Query Parameterization Cheat Sheet
- OWASP Command Injection Article
- OWASP XML eXternal Entity (XXE) Reference Article
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing

External

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection

What makes assuring Security so difficult?

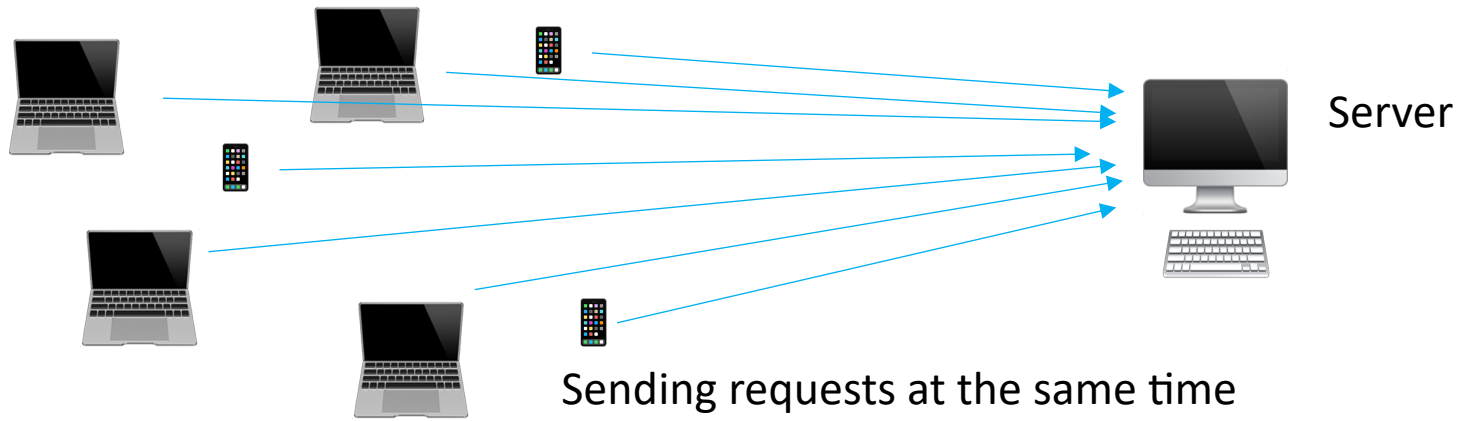
- Security is a system property!
- Vulnerabilities can be at different places in the system
- Systems rely on other systems
- Search for the unknown (a vulnerability that can be exploited!)

Corporate Concerns & Individual risks

Corporate concerns

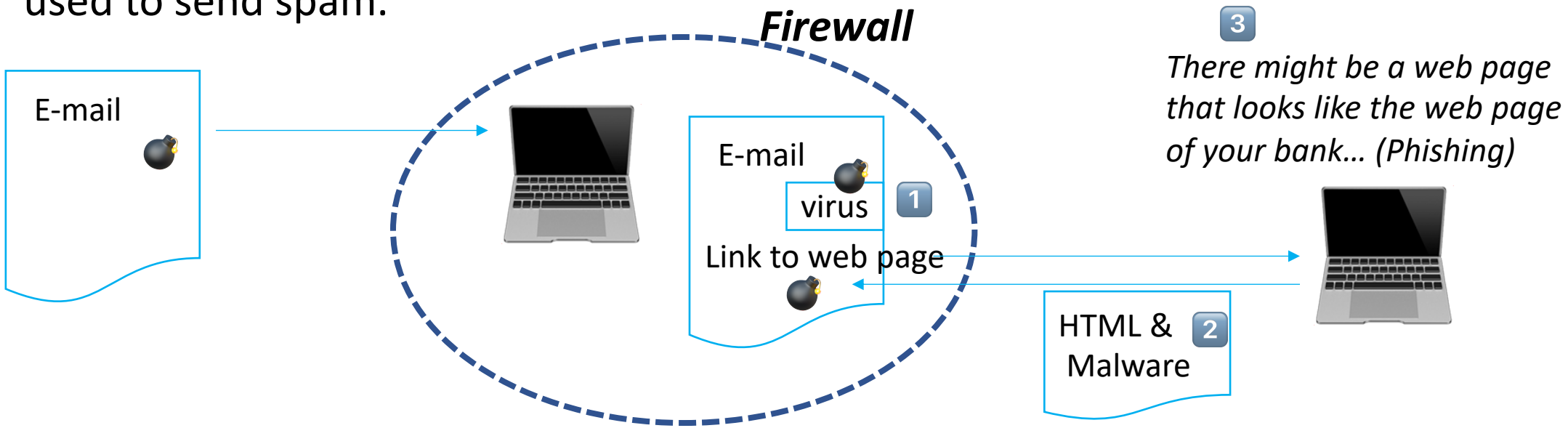
- **Denial of Service**:
  - Actions that make access to services not available
  - 2 types of attacks:
    - **Bandwidth consumption**: The intruder tries to consume all available bandwidth between the server and the public internet to prevent from accessing the server
    - **Exceeding server resources**: The intruder tries to overload the processing power of a server, e.g., by sending a lot of requests at the same time.

DoS Attack



Corporate concerns

- **Unsecured E-mail**
  - E-mails may include links to web pages comprising malware or viruses themselves.
  - E-mail server passwords may be easy to hack and the e-mail system can be used to send spam.



Corporate concerns

- **Online Fraud Risk**:
  - Using someone else e.g. credit card data to purchase orders
  - ***Business responsibility*** — Businesses must repay credit card companies for illegal purchases. Businesses must also assure that customers' credit card details cannot be stolen.
  - ***Falling sales*** — Consumers will stay away from specific merchants that have received bad press about fraudulent incidents.

  - **Prevention**:
    - **Geolocation**: risk based on geographical location of an IP address regarding frauds
    - **Order velocity monitoring**: monitor usage of cards and report on unexpected behavior (like too many orderings at the same time at different places)

Corporate concerns

- **Information handling practices**
  - **Information acquisition** – Which information is stored and why?
  - **Storage Data access** – Is the information stored secure? What data access restrictions are applied?
  - **Database** – Is access audited and password controlled?
  - **Data disposal** – How and when is information deleted?
  - **Personnel** – Access restrictions? Regular background checks for personnel having access to confidential or otherwise restricted information.

Has also to do with privacy concerns of customers!!!

Corporate concerns

- **Port Scanning and Back Door Access**:
  - Searching for port addresses on side of computers that can be accessed!
  - **Note**: ports are specific addresses in a computer used to communicate, e.g., port 80 is for http, 20 is for ftp, 156 for SQL servers, and 443 for https.

  - If ports are open they can be used to access your computer from outside! (It is like leaving a window open for others to come in)

Individual risks

- **Cookies** used for web pages
- **Viruses** and **worms**
- **Keystroke Logging**
- **Phishing** is an attempt to trick a person into giving away their private account information by confirming it at the phisher's Web site.
- **HTML** code in **emails** (may not be visible but can take actions)
- **Spam**
- **Identity theft**
- **Password hacking**: from brute force to social engineering

Protection measures

Protection measures

- **Firewalls**: Hardware/Software that prevents forbidden communication. Can be seen as a filter between the computer and the internet that allows only IP packets that are allowed passing the firewall.
  - IP address filtering
  - Port blocking
  - Content filtering

- **Intrusion detection and prevention systems**: Identify forbidden communication and take actions against such events.

Protection measures

- **Anti Virus software**
  - Identifies viruses because of
    - Behavior
    - Similar signatures (parts of the virus known already)
- **Anti Spyware software**
  - Real-time protection preventing the downloading and installation of spyware.
  - Detection and removal of spyware as the hardware (memory and secondary storage) is scanned.
- Using **digital signatures** to allow
  - Author identification
  - Content integrity

Protection measures

- Use of **Sandboxes** for running programs providing (e.g., Docker)
  - A tightly-controlled environment including protected disk and memory resources.
  - No or limited access to networks
  - No or limited access to the file system
- **Backup and Recovery**
  - Disaster recovery
  - Data recovery
- Use of **strong passwords**

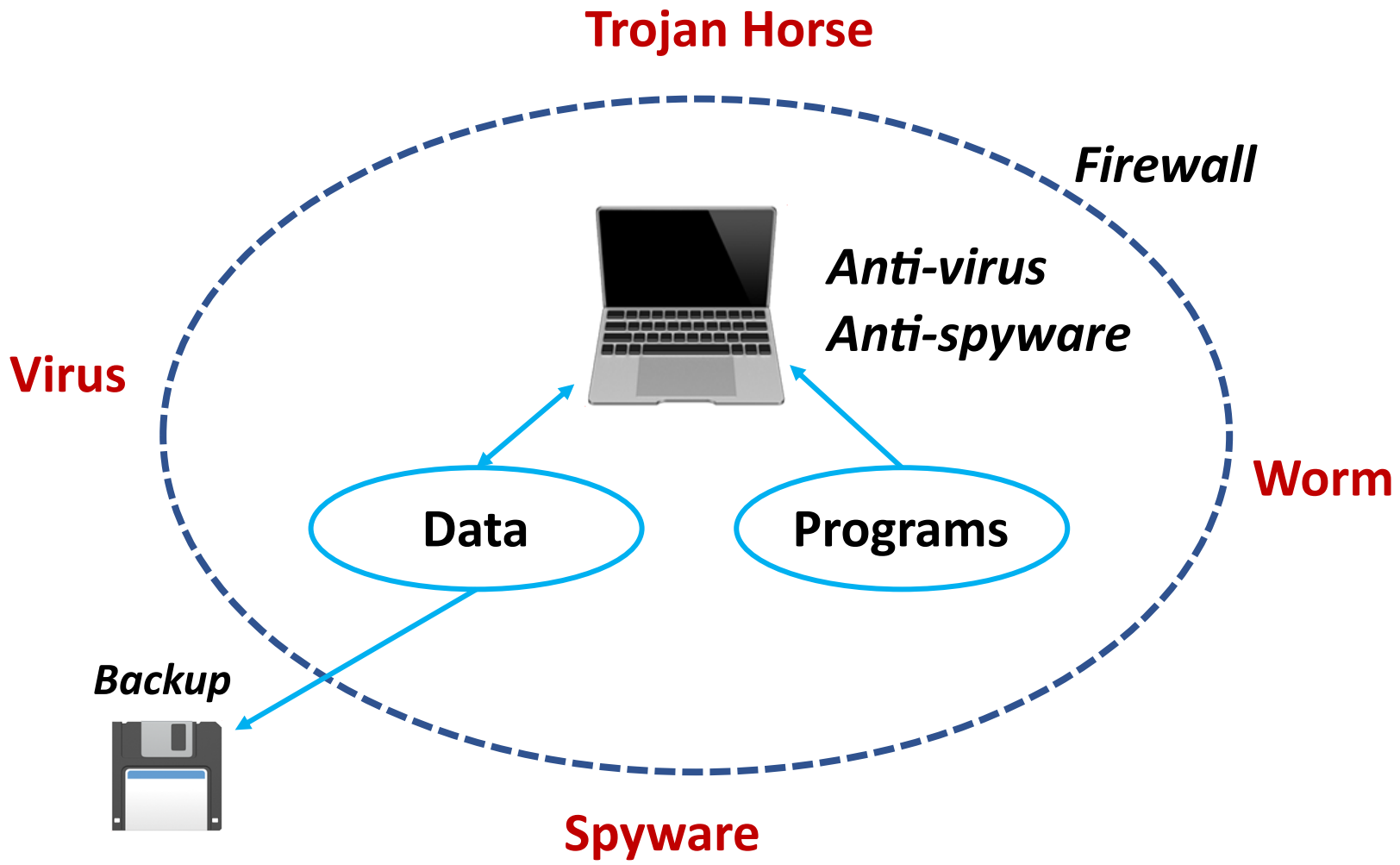
Protection measures

- Use of **encryption**:
  - **Symmetric-key** – Each computer has a key (a secret code) that is used for encoding and decoding messages. The same key is used by both computers.
  - **Public-key** – This method uses a combination of a private key and a public key. The private key is known only to one computer, while the public key is given to any computer that wants to communicate securely with it. To decode an encrypted message, a computer must use the public key, provided by the originating computer, and its own private key. Public-key encryption is commonly used in computer networks.
  - In cryptography, **plaintext** is information used as input to an encryption algorithm; the output is termed **ciphertext**.

Protection measures

- Use of **uninterruptible power supply** (UPS) to prevent from physical damage and data loss.
- **Physical security**:
  - Lock computer equipment from unauthorized access!
- **Strong authentication schemes**:
  - Strong passwords larger than 8 characters comprising capitalized characters, numbers and special characters like &, %, $,…
- **Biometrics**: Fingerprints, Retina scans, …

Security models



But there is always the risk from insight too!

Ethics and Privacy

Why ethics and privacy?

- Information technology affects fundamental rights involving
  - copyright protection,
  - intellectual freedom,
  - accountability, and
  - security.

- **Code of Ethics**
  - Set of rules and conventions that are commonly agreed to within a business, academic, or governmental organization.
  - Why?
    - To define accepted and acceptable behaviors.
    - To promote high standards of practice.
    - To provide a benchmark for members to use for self evaluation.
    - To establish a framework for professional behavior and responsibilities.
    - As a vehicle for occupational identity.

ACM Code of Ethics and Professional Conduct

Preamble

Computing professionals' actions change the world. To act responsibly, they should reflect upon the wider impacts of their work, consistently supporting the public good. The ACM Code of Ethics and Professional Conduct ("the Code") expresses the conscience of the profession.

The Code is designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way. Additionally, the Code serves as a basis for remediation when violations occur. The Code includes principles formulated as statements of responsibility, based on the understanding that the public good is always the primary consideration. Each principle is supplemented by guidelines, which provide explanations to assist computing professionals in understanding and applying the principle.

Association for Computing Machinery (ACM) Code of Ethics

- Contribute to society and to human well-being.
- Avoid harm to others.
- Be honest and trustworthy.
- Be fair and take action not to discriminate.
- Honor property rights including copyrights and patent.
- Give proper credit for intellectual property.
- Respect the privacy of others.
- Honor confidentiality.