Secure Coding: Web Feb 18th, 2026

Working class notes for CNT 4419: Secure Coding the the University of South Florida.

Trevor Flahardy

目录

Part 1: Mechanisms

Types (categories) of mechanisms:

- Prevent
- Detect
- Contain
- Recover

The professor notes that these categories are not "crystal clear"; they can be subjective depending on the context. So, for example, a **dynamic type checker** would fall as one of these things. It's either preventative or detective, depending on how you look at it (eager or lazily checking types).

1.1. Preventative Mechanisms

Done before something bad has happened. Examples of this are such as a firewall (FW) and passwords.

A static type checker would be a preventative mechanism (an eager one that checks types at compile time).

1.2. Detective Mechanisms

Detective mechanisms are **done after something bad has happened**.

An example of this would be performing audits and monitoring. This is done to detect if something bad has happened (so taking logs and analyzing them).

Some aspects of anti-virus software are detective mechanisms, such as the signature-based detection. This is because it detects if a virus is present on the system.

1.3. Containment Mechanisms

The idea here is that: let's assume an attack has been successful and we want to contain it such that it has the smallest effect possible on our system. So how we do that?

The trick is **replication**, so we can kind of "forget about" one machine and continue with our operations on another machine because we have replicated the capabilities of the first machine on the second machine. So if the first machine gets compromised, we can just "throw it away" and continue with the second machine.

This is the big idea: **replication** with containment mechanisms.

The main idea here is that you are trying to **isolate** a problem. Ideally, the machine that has been compromised is isolated and **simply can be shut off**. Additionally, instead of powering the system off you can isolate it.

1.4. Recovery Mechanisms

Revert to backup. This may be backup data, or a backup system(s) or machine(s); but somehow you are going to revert to a known good state.

So a common way to recover something is to cut and restore power (turn off and back on again); using backed up data on another machine, etc.

When you look at this for devastating attacks (when the attacker has been very successful), performing a restart may not be enough. So you may have to re-flash things or switch machines entirely.

The professor notes that another way to recover is to "file an insurance claim".

Aside Note

So how we do we know if some software is malicious (aka malware)? It turns out that attackers have a lot of sophisticated tricks too make their software look "legit" (or safe). So once some malware is discovered and people want to start preventing it, malware writers circumvent this by adapting how the malware looks or behaves. This is known as **polymorphic**. So how we do we know if some software is malicious (aka malware)? It turns out that attackers have a lot of sophisticated tricks too make their software look "legit" (or safe). So once some malware is discovered and people want to start preventing it, malware writers circumvent this by adapting how the malware looks or behaves. This is known as **polymorphic**.

> 🧠 **Definition**
>
> **Polymorphic malware**: Malware that can take many shapes or forms.

- There are also some interesting contexts in which we can associate these goals with "medical" terminology:
  - Preventing some disease
  - Detecting some disease
  - Containing some disease (someone who has a disease)
  - Recovering from some disease (getting better from a disease)

Often people like to make these analogies between disease and medical contexts and security contexts. So for example, we can think of a virus as a disease, and then we can think of anti-virus software as a way to prevent or detect that disease. We can also think of a firewall as a way to prevent unauthorized access to our system, similar to how we might use a vaccine to prevent a disease.

> ❓ **Question**
>
> **But why is that a difficult analogy to make?**
>
> As you get into the deep aspects of computer security, measuring "system health" and similar metrics becomes very difficult. How do you know a security mechanism is good? Or that it is preventing the attacks we want it to prevent?

So associate this idea to medicine. Say 1,000 people have some new disease and you give some new drug to all of them and everyone is cured. This would be a resounding success and the disease is resolved. Switch over to computer security and take 1,000 machines with some malware on it. Some mechanism has been put in place, and the mechanism detects or cleans up after all 1,000 instances of that attack. Can we claim that we have "solved" this attack? Maybe this specific attack, **but what about polymorphic versions of it?** So a very intelligent adversary may change their program very quickly, or automatically, to go around your mechanism. So as fast as you can change a mechanism is as fast as an attacker can change it.

1.4.1. Keeping "Good" Backups

Ransomware: is an example of an attack that is devastating and can be prevented by having good backups. So the idea is that you have some malware that encrypts your data and then demands a ransom to give you the decryption key.

- The defense against them are replication, revering to your backup, getting a new machine, reverting to factory settings, etc. So the idea is that you have a backup of your data and you can restore it if something bad happens.
- There is often a *trust among thieves*: If you pay the ransom, the attacker may give you the decryption key. However, there is no guarantee that they will actually do so, and they may even demand more money after you pay the initial ransom. So it's a risky situation to be in. This is the attacker's "business" model, and it's within the attacker's best interest to do what they say (providing the key upon payment) to continue to make money from other victims.

There exists a more modern variation on ransomware, though. The ransomware may not just encrypt your data ("scramble" your data) - now often the ransomware will **leak your data**. This is called **exfiltrate** your data or sensitive information.

- Commonly, all the data on the machine is exfiltrated (including PII information).

> 🧠 **Definition**
> **Exfiltration**: The unauthorized transfer of data from a computer or network. This is often done by attackers to steal sensitive information, such as personal identifiable information (PII), financial data, or intellectual property.

> 🧠 **Definition**
> **Personal Identifiable Information (PII)**. This is information that can be used to identify an individual, such as their name, address, social security number, etc. This is often the type of data that is exfiltrated in a ransomware attack.

1.5. Mechanism: Adhere to Principle of Least Privilege (PoLP)

1.6. Mechanism: "Avoid Security by Obscurity"

This is the idea that you should not rely on secrecy of your design or implementation as the main method of providing security. So for example, if you have some software that has a vulnerability, and you try to hide that vulnerability by not disclosing it, this is not a good security practice. This is because attackers can still find the vulnerability through other means, such as reverse engineering or fuzzing.

Typically, **we want to try and encapsulate any secrets being relied on into cryptographic keys**.

1.7. Mechanism: Be careful about when and where/how keys are stored

- So we ant to isolate these secure things into "keys", and then we want to be really careful about where we store these keys. We want to make sure that they are not stored in a way that is easily accessible to attackers, such as in plain text on a hard drive or in a configuration file.
- Most hardware, now, has **special hardware that helps us protect these keys**. Processors or modules are specifically created for this purpose
  - Called the **TTM Trusted Platform Module** (at least, on Windows)

- The OS often has APIs for using this part of the machine. Your code can call the APIs and make use of this hardware.

So making these truly random keys is very difficult. One very popular one is the Cloudflare lava lamp room.

This is a room full of lava lamps that are randomly moving around, and the patterns of the lava lamps are used to generate random numbers. This is a very creative way to generate random numbers, and it is also very secure because it is based on physical randomness.

Historically, it's very difficult to write your own "random number generator" – the professor recommends using a well-known library for this.

🧠 **Definition**

Cryptographically Secure Pseudo Random Number Generator (CSPRNG): A CSPRNG is a type of random number generator that is designed to be secure against certain types of attacks. It is often used in cryptographic applications to generate keys, nonces, and other random values that need to be unpredictable and resistant to reverse engineering.