Attack patterns

- In computer science, **attack patterns** are a group of rigorous methods for finding bugs or errors in code related to computer security.
- Attack patterns are often used for testing purposes and are very important for ensuring that potential vulnerabilities are prevented. The attack patterns themselves can be used to highlight areas which need to be considered for security hardening in a software application. They also provide, either physically or in reference, the common solution pattern for preventing the attack. Such a practice can be termed *defensive coding patterns*.
- Attack patterns define a series of repeatable steps that can be applied to simulate an attack against the security of a system.

- There are several different ways to categorize attack patterns. One way is to group them into general categories, such as:
- Architectural, Physical, and External .
- Another way of categorizing attack patterns is to group them by a specific technology or type of technology (e.g. database attack patterns, web application attack patterns, network attack patterns, etc. or SQL Server attack patterns, Oracle Attack Patterns, .Net attack patterns, Java attack patterns, etc.)

- **Using General Categories**
- **Architectural** attack patterns are used to attack flaws in the architectural design of the system. These are things like weaknesses in protocols, authentication strategies, and system modularization. These are more logic-based attacks than actual bit-manipulation attacks.
- **Physical** attack patterns are targeted at the code itself. These are things such as SQL injection attacks, buffer overflows, race conditions, and some of the more common forms of attacks that have become popular in the news.
- **External** attack patterns include attacks such as trojan horse attacks, viruses, and worms. These are not generally solvable by software-design approaches because they operate relatively independently from the attacked program. However, vulnerabilities in a piece of software can lead to these attacks being successful on a system running the vulnerable code.
- An example of this is the vulnerable edition of Microsoft SQL Server, which allowed the Slammer worm to propagate itself.[1] The approach taken to these attacks is generally to revise the vulnerable code.

Structure of attack patterns

- Attack Patterns are structured very much like structure of Design patterns. Using this format is helpful for standardizing the development of attack patterns and ensures that certain information about each pattern is always documented the same way.
- A recommended structure for recording Attack Patterns is as follows:

- **Pattern Name**
- **Type & Subtypes:**Typical types include Injection Attack, Denial of Service Attack, CryptanalysisAttack, etc. Examples of typical subtypes for Denial Of Service for example would be: DOS – Resource Starvation, DOS-System Crash, DOS-Policy Abuse.
- **Also Known As**
- **Description**
- **Attacker Intent**
- **Motivation**
- **Exploitable Vulnerability**
- **Participants**
- **Process Diagram**
- **Dependencies and Conditions**
- **Sample Attack Code**
- **Existing Exploits**
- **Follow-On Attacks**
- **Mitigation Types**
- **Recommended Mitigation**

- **Pattern Name**
- The label given to the pattern which is commonly used to refer to the pattern in question.
- Types and subtypes:
- Typical types include Injection Attack, Denial of Service Attack, CryptanalysisAttack, etc. Examples of typical subtypes for Denial Of Service for example would be: DOS – Resource Starvation, DOS-System Crash, DOS-Policy Abuse.
- **Also Known As**
- Certain attacks may be known by several different names. This field is used to list those other names.

- **Description**
- This is a description of the attack itself, and where it may have originated from. It is essentially a free-form field that can be used to record information that doesn't easily fit into the other fields.
- **Attacker Intent or motivation**
- This field identifies the intended result of the attacker. This indicates the attacker's main target and goal for the attack itself. For example, The Attacker Intent of a DOS – Bandwidth Starvation attack is to make the target web site unreachable to legitimate traffic.

- This field records the attacker's reason for attempting this attack. It may be to crash a system in order to cause financial harm to the organization, or it may be to execute the theft of critical data in order to create financial gain for the attacker.
- **Exploitable Vulnerability**
- This field indicates the specific or type of vulnerability that creates the attack opportunity in the first place. An example of this in an Integer Overflow attack would be that the integer based input field is not checking size of the value of the incoming data to ensure that the target variable is capable of managing the incoming value. This is the vulnerability that the associated exploit will take advantage of in order to carry out the attack.

- **Participants**
- The Participants are one or more entities that are required for this attack to succeed. This includes the victim systems as well as the attacker and the attacker's tools or system components. The name of the entity should be accompanied by a brief description of their role in the attack and how they interact with each other.
- **Process Diagram**
- These are one or more diagrams of the attack to visually explain how the attack is executed. This diagram can take whatever form is appropriate but it is recommended that the diagram be similar to a system or class diagram showing data flows and the components involved.

- **Dependencies and Conditions**

Every attack must have some context to operate in and the conditions that make the attack possible. This section describes what conditions are required and what other systems or situations need to be in place in order for the attack to succeed. For example, for the attacker to be able to execute an Integer Overflow attack, they must have access to the vulnerable application. That will be common amongst most of the attacks. However, if the vulnerability only exposes itself when the target is running on a remote RPC server, that would also be a condition that would be noted here.

- **Sample Attack Code**

If it is possible to demonstrate the exploit code, this section provides a location to store the demonstration code. In some cases, such as a Denial of Service attack, specific code may not be possible. However, in Overflow, and Cross Site Scripting type attacks, sample code would be very useful.

- **Existing Exploits**

Exploits can be automated or manual. Automated exploits are often found as viruses, worms and hacking tools. If there are any existing exploits known for the attack this section should be used to list a reference to those exploits. These references can be internal such as corporate knowledge bases, or external such as the various CERT, and Virus databases.

Exploits are not to be confused with vulnerabilities. An Exploit is an automated or manual attack that utilises the vulnerability. It is not a listing of a vulnerability found in a particular product

- **Follow-On Attacks**

Follow-on attacks are any other attacks that may be enabled by this particular attack pattern. For example, a Buffer Overflow attack pattern, is usually followed by Escalation of Privilege attacks, Subversion attacks or setting up for Trojan Horse / Backdoor attacks. This field can be particularly useful when researching an attack and identifying what other potential attacks may have been carried out or set up.

- **Mitigation Types**

The mitigation types are the basic types of mitigation strategies that would be used to prevent the attack pattern. This would commonly refer to Security Patterns and Defensive Coding Patterns. Mitigation Types can also be used as a means of classifying various attack patterns. By classifying Attack Patterns in this manner, libraries can be developed to implement particular mitigation types which can then be used to mitigate entire classes of Attack Patterns. This libraries can then be used and reused throughout various applications to ensure consistent and reliable coverage against particular types of attacks.

- **Recommended Mitigation**

Since this is an attack pattern, the recommended mitigation for the attack can be listed here in brief. Ideally this will point the user to a more thorough mitigation pattern for this class of attack.