e-ISSN : 2231-2471, p-ISSN : 2319-2224 © VSRD International Journals : www.vsrdjournals.com

RESEARCH COMMUNICATION

USING EXPLOIT PATTERNS TO DEVELOP SECURE SOFTWARE

[1]Rizwan Khan* and [2]Santosh Kumar
[1]Assistant Professor, ABESIT Group of Institution. Ghaziabad UP INDIA.
[2]Professor Department of Mathematics, The University off DODOMA, TANZANIA.
*Corresponding Author : rijwankhan786@gmail.com

ABSTRACT

Building secure software becomes more and more challenging every day because of the increasing size, complexity of products and the increasing number and skills of the attackers. To develop secure software, the developer must ensure that every possible and potential vulnerability has been protected as it takes only a small unprotected vulnerability for the attacker to bring a system to its knees. To be effective, the developers must have a firm grasp of the attacker's perspective and his methodologies and resources. Repositories like CERT, MITRE, BSI etc which record and analyze the exploit incidents and provide services like vulnerability databases are an excellent aid for the developers. This paper introduces exploit patterns and discusses their use as an important method of empowering developers to develop secure software. The exploit patterns can bring considerable value for software security considerations through all phases of SDLC.

Keywords : Vulnerabilities, Attacks, Exploits, Exploit Patterns.

1. INTRODUCTION

The Challenge : The primary challenge in building software with adequate level of security fitted in is that it is much easier to find vulnerabilities in software than it is to make software secure, just as it is easy to find a way to break a safe then to guarantee that a safe is break-proof. Moreover, the virtuality of software adds to the complexity of the problem. With many systems, the attacker may actually possess the software (it is quiet easy to obtain a local copy to attack) or could attack it remotely through networks. With the ability to attack remotely and without physical access, attacks become much easier and preventing them becomes much difficult. Audit trails, generated by systems to record activities may not be sufficient to catch attackers after an attack takes place, because attackers could leverage the anonymity of an unsuspecting user's wireless network or public computers to launch attacks.

Another challenge faced by developers is that of market forces that demand software developers to maximize functionality and minimize time to market. Functionality is what generally sells software, and security is usually treated as a stepsister, an afterthought. Because most security capabilities are transparent to the users, they are not usually considered a priority. The products offering the best functionality are unfortunately often chosen over the ones that offer the best security. Because of this, more and more systems are being exploited with significant and devastating consequences. As time passes and as the industry is becoming wiser to the ways of the attackers and to the potential loss from these attacks, the shortsightedness of this approach is becoming clear to the industry, but it will still remain a challenge for many years to come.

Another major challenge is the lack of knowledge. The attacker community is already well-versed in the technologies and methodologies used to exploit weaknesses and attack software, but the software development community is not generally educated in the wicked ways in which software is exploited. Such knowledge is still mostly considered unnecessary and one trying to get this knowledge is usually not viewed favorably.

This paper is organized as: - section 2 describes the exploit patterns in some detail, section 3 presents a

Proposed framework structure along with a brief description of the fields; section 4 describes the various ways in which an exploit pattern can be used.

2. THE SOLUTION – EXPLOIT PATTERNS

One potential solution to these challenges is to broaden the view of developers by providing then knowledge that enables them to understand the attacker's perspective. We are all familiar with Design patterns which are a familiar tool used by the software development community to help solve recurring problems encountered during software development These patterns help us to develop secure, stable, and effective software architecture and design. Exploit patterns apply the problem-solution paradigm of design patterns in a destructive rather than constructive context. Exploit patterns describe the techniques that attackers may use to break software.

Technically speaking An Exploit pattern is A series of repeatable steps that can be applied in a consistent and reliable manner to simulate an attack against a software

system. Exploit patterns are a useful medium to classify and categorize attacks in a meaningful way, such that problems and solutions can be discussed effectively. They provide composed and concise information about the relevant vulnerabilities, exploits and mitigation methods. Instead of taking an ad-hoc approach to software security, exploit patterns can identify the types of known attacks to which an application could be exposed so that mitigations can be built into the application.

Goals and Benefits of Exploit Patterns

- The rationale behind using exploit patterns is that software developers must think like attackers to anticipate threats and thereby effectively secure their software.
- The concept of exploit patterns can be used to teach the software development community how attackers go about exploiting the software in reality and to implement proper ways to avoid the attacks. Exploit patterns provide a way for software developers to learn about how their software may be attacked. Armed with knowledge about possible or probable attacks, developers can take steps to mitigate the likelihood or impact of these attacks.
- The goal of an Exploit Pattern is to represent the attacker's perspective in a formalized and constructive way: to provide expert-level understanding and guidance to software development personnel of all levels as to how their software is likely to be attacked, and thereby equip them to build more secure software
- An exploit pattern is a blueprint for an exploit. It is a description of a common approach attackers take to attack software. They are developed by reasoning over large sets of software exploits and attacks.
- Exploit patterns help identify and qualify the risk that a given exploit will occur in a software system. Exploit patterns are often used for testing purposes and are very important for ensuring that potential vulnerabilities are prevented. The exploit 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.
- Often, security policy also lacks a comprehensive understanding of the issues surrounding software security, as developers have a natural propensity to think in terms of features and functions. Exploit patterns also lend to bridge this gap.
- Another benefit of exploit patterns is that they contain sufficient detail about how attacks are carried out to enable developers to help prevent them. Exploit patterns, however, do not typically contain inappropriately specific details about the actual exploits to ensure that they do not help educate less skilled members of the black hat community (e.g, script kiddies). Information from exploit patterns generally cannot be used directly to create automated exploits.

Categories : From the development life cycle end of the spectrum there are two categories of exploit patterns, Architectural and Physical.

- Architectural exploit patterns are used to attack weaknesses or vulnerabilities in the architectural design of the system. These may include attacks against a weaknesses in protocols, authentication strategies, and system modularization. These are more logical level attacks than actual bit manipulation attacks.
- Physical exploit patterns are targeted at the code itself. These are things such as SQL Injection attacks, Buffer Overflows, race conditions, and the more common forms of attacks that have become popular in the news and with script kiddies. These target the vulnerabilities of the programming languages which inadvertently provide attackers a back door to enter the code.
- There is a third category of exploit patterns that are independent such as Trojan Horse attacks, viri, and worms. These are not something that you can design out of your product. They operate outside and independent of any particular product. However vulnerabilities in a product can facilitate these other attacks or increase their chances of being successful, or at a minimum facilitating these attacks on a system hosting the vulnerable product. For example the vulnerable edition of SQL Server which allowed the Slammer worm to propagate itself. The worm itself can't be designed out, but the vulnerability in the product can be.

3. STRUCTURE OF AN EXPLOIT PATTERN

An exploit pattern is an abstraction mechanism for describing how a type of observed attack is executed. Following the pattern paradigm, it also provides a description of the context where it is applicable and then, unlike typical patterns, it gives recommended methods of mitigating the attack. Thus, an exploit pattern is a blueprint for an exploit minus the actual code needed for the attack itself. In order to accurately and succinctly describe an Exploit pattern, we must have a consistent and easily consumable way of representing the pattern to the user. Only then can we have a uniform platform where exploits reported at a variety of forums or databases can be converted to standard exploit patterns to be used by the developer community at large. Thus our approach is to try and stay close to existing pattern representations that exist in the industry and add the necessary information to ensure we convey the pattern properly. This will allow for easy adoption as the format is familiar. However our new Exploit pattern structure contains different fields, and they are used quite differently than standard software patterns.

Fields of the Proposed AP structure

Table 3.1 : Proposed Exploit Pattern Format

Fields of the Proposed AP structure	
1. Pattern Name and Classification	2. Types and Subtypes
3. Also Known As	4. Attack Prerequisites
5. Description	6. Attacker Motivation and Results
7. Related Exploitable Vulnerability	8. Method of Attack
9. Process Diagram	10. Mitigation Types

Exploit pattern Fields : Table 3.1 shows a proposed structure for recording Exploit patterns

- **Pattern Name & Classification :** A unique descriptive identifier.
- **Types & Subtypes :** The pattern type and its associated subtypes aid in classification of the pattern. This allows users to rapidly locate and identify pattern groups that they will have to deal with in their security efforts.
- **Also Known As :** Certain attacks may be known by several different names.
- **Attack Prerequisites :** Every attack must have some context (Platform, OS, language, architectural paradigm) to operate in and the conditions that make the attack possible. This describes what conditions are required and what other systems or situations need to be in place for the attack to succeed. For example, for an Integer Overflow attack to succeed , the attacker must have access to the vulnerable application.
- **Description :** A description of the attack including the chain of actions taken.
- **Attacker Motivation and Results :** The intended result of the attacker. Here we indicate 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.
- **Related Exploitable Vulnerability or Weakness**: 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. Unchecked inputs, outputs, weak cryptography, weak authentication are examples of types of exploitable vulnerabilities. Specific vulnerabilities should reference industry standard identifiers such as Common Vulnerabilities and Exposures (CVE) number, US-CERT number, etc. Specific weaknesses (underlying issues that may cause vulnerabilities) should reference industry-standard identifiers such as the Common Weakness Enumeration (CWE).
- **Method of Attack :** What is the vector of attack used (e.g., malicious data entry, maliciously crafted file, protocol corruption)?
- **Process Diagram :** This is 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.
- **Mitigation Types :** the basic types of mitigation strategies that would be used to prevent the exploit pattern. This would commonly refer to Security Patterns and Defensive Coding Patterns. Mitigation Types can also be used as a means of classifying various exploit patterns. By classifying Exploit patterns in this manner, libraries can be developed to implement particular mitigation types which can then be used to mitigate entire classes of Exploit patterns.
- **Resources Required :** What resources (e.g., CPU cycles, IP addresses, tools, time) are required to execute the attack?
- **References :** What further sources of information are available to describe this attack?

4. EXPLOIT PATTERN USAGE

Both application development and standard IT operations can benefit from the use of Exploit patterns. Exploit patterns can also be used by educators to teach future developers common problems faced by applications and introduce them to good secure development practices in a consistent and predictable fashion.

Exploit patterns in Software Development : Exploit patterns can be used during almost all phases of SDLC. During the design phase exploit Patterns help to:

- Determine attacks the application is likely to face.
- Plan mitigation strategies for the identified attacks and provide developers with mitigation guidance
- Provide testers with test guidance to ensure that the various attacks have been mitigated
- Increase the security knowledge and awareness of the development team

When the application is in the early stages of conception Exploit Patterns can be used to identify potential risks to the application, and the likelihood of those risks being realized.

A second way in which Exploit Patterns can help is by providing common consistent unit testing techniques that developers can implement during their development phase. This will help ensure that the various identified exploits that the application is designed to mitigate are taken care of in the development phase rather than the problems being

discovered after the application has been released to test. EPs provide testers with guidance on the attacks the application is designed to face, as well as a means of checking to see that the mitigation strategies have been implemented.

EPs provide clear and consistent testing points for testers to be able to apply against an application to ensure that the mitigation strategies are implemented and working as they should be. This will increase the confidence in the test cycle, as well as in deployment, that the application has at least had some kind of security evaluation performed against it before it goes in to the wild.

Exploit patterns in IT Operations : When applications are deployed and become part of normal IT operations EPs can aid the IT Operations team in several areas.

- Provide some guidance in system testing
- Assist in identifying possible attacks to the system
- Provide security knowledge and awareness to the operations team

Once applications are deployed, they need to periodically be checked to ensure that they are not subject to any new or unknown types of attacks.

Exploit pattern Libraries (EPLs) are living documents that are constantly being updated. This provides an IT operations team with the ability to review the APL and test their systems against new or existing attacks.

Exploit patterns in Education : Exploit patterns can be used in education in order to teach students how applications are attacked, and what kinds of techniques they will need to employ in order to prevent their applications from being vulnerable to those kinds of attacks.

Exploit patterns provide a mechanism by which students can be introduced to the attack and its mitigation together. This provides a link between the two that the student can focus on, and use in their projects and future development efforts. Since EPs link the attack, the mitigation, the testing strategy and reference material in one unit, students will have a wealth of information in a centralized location. This is a tool that they can used to learn defensive security practices alongside their normal development techniques. Exploit patterns should be blended with risk management and risk assessment techniques in order to get maximum benefit.

5. CONCLUSION

This paper provides an introductory glimpse at the concept of Exploit Patterns as a major tool in assisting development of secure software. Exploit Patterns can be helpful in providing extensive attack prevention help to developers at a common location which can be used by the software development community at large to ensure that their software are well protected against at least the known exploits and vulnerabilities.

A lot of work needs to be done in this area, especially in the direction of creation of Exploit patterns. Ways and means have to be developed to generate these patterns automatically from the information available at repositories like CERT, MITRE etc. AI based technologies like intelligent agents can play an important role in this direction.

6. REFERENCES

[1] Gamma, E.; Helm, R.; Johnson, R.; & Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston, MA: Addison-Wesley, 1995.
[2] Graff, Mark & van Wyk, Kenneth. Secure Coding: Principles and Practices. Sebastopol, CA: O'Reilly and Associates, 2003
[3] Gegick, Michael & Williams, Laurie. "Matching Attack Patterns to Security Vulnerabilities in Software-Intensive System Designs." ACM SIGSOFT Software Engineering Notes, Proceedings of the 2005 work-shop on Software engineering for secure systems—building trustworthy applications SESS '05, Volume 30, Issue 4. New York, NY: ACM Press, 2005.
[4] Schneider, Bruce. "Attack Trees: Modeling Security Threats." Dr. Dobb's Journal, December, 1999.
[5] Computer Emergency Readiness Team www.cert.org
[6] www.buildsecurityin.us-cert.gov
[7] http://capec.mitre.org

●●●