**OWASP** (Open Web Application Security Project) е глобално призната организация, посветена на подобряването на софтуерната сигурност. Тя предоставя безплатни ресурси, инструменти и образователни материали, които подпомагат разработчиците и специалистите по киберсигурност в защитата на приложенията. Един от ключовите приноси на OWASP е известният списък **OWASP Top 10**, който излага десетте най-сериозни рискове за сигурността на приложенията. Решаването на тези рискове е от критично значение за подобряването на защитата на чувствителните потребителски данни и за предотвратяване на потенциални атаки.

В този реферат ще бъде разгледан един от най-значимите рискове в този списък – **инжекционните атаки (Injection)**. Този тип атаки, макар и стари, продължават да представляват сериозна заплаха за уеб приложенията. Те са класирани като най-голям риск през 2017 година и заемат трето място през 2021. Инжекционните атаки се отличават с манипулиране на некоректно подадени входни данни, които се обработват от интерпретатор, водейки до промяна в поведението на приложението. Това може да доведе до сериозни последствия като кражба на данни, срив на системата или отказ на услуга. Основната причина за възникването на този риск е недостатъчната валидация на входните данни, което го прави лесно достъпен за неопитни нападатели.

Приложение е уязвимо към този вида атака , ако:

Потребителските данни не се валидират, филтрират или почистват от приложението.

Динамични заявки или не параметризирани извиквания без контекстуално избягване се използват директно в интерпретатора.

Враждебни данни се използват в параметри за търсене в ORM (Object-Relational Mapping), за да се извлекат допълнителни и чувствителни записи.

Враждебни данни се използват директно или се конкатенират. SQL или командата съдържат структурата и злонамерените данни в динамични заявки, команди и съхранени процедури.

**Неправилна валидация на входни данни**

Продуктът приема входни данни, но не ги валидира правилно или изобщо не извършва валидиране, което води до опасности за сигурността.

Освен валидирането, други техники като филтриране (премахване на опасни елементи) или кодиране (предотвратяване на неправилно интерпретиране на данни) също са ефективни за обработка на входни данни.

#### **Видове данни за валидиране:**

* **Сурови данни**: низове, числа, параметри, файлово съдържание.
* **Метаданни**: информация за данните, като заглавки (headers) ,размер и локация.

#### **Основни свойства за валидиране:**

* Размер, дължина, честота.
* Правилно форматиране и синтаксис.
* Тип данни (реален или очакван).
* Съответствие с бизнес логика или специфични правила.
* Автентичност, собственост.

### **Неправилно неутрализиране на специални елементи в изходни данни, използвани от компоненти по веригата (CWE 74)**

Този тип уязвимост възниква, когато продуктът конструира команда, структура от данни или запис, използвайки входни данни, повлияни от външни източници, но не неутрализира правилно специалните елементи. Тези елементи могат да променят начина, по който данните се интерпретират или обработват от компонентите по веригата.

**Ключови характеристики на уязвимостите от тип инжектиране:**

* Те позволяват инжектиране на контролни данни в потребителския поток от данни.
* Изпълнението на процеса може да бъде променено чрез изпращане на код през легитимни канали за данни, без да е необходима допълнителна уязвимост.
* Инжекционните атаки се различават от други видове, като например **препълване на буфер**, тъй като изискват само правилно разчитане и обработка на подадените данни.

### **Примерни сценарии:**

* **SQL Injection:** Въвеждане на злонамерени SQL команди чрез входни данни.
* **Command Injection:** Инжектиране на команди към операционната система чрез уеб интерфейс или API.

#### **Предотвратяване**

* **Неутрализиране на специални символи:** Използвайте екраниране, кодиране или други техники за предотвратяване на неправилно интерпретиране на данните.
* **Използване на параметризирани заявки:** Особено при взаимодействие с бази данни.
* **Проверка на входните данни:** Ограничаване на допустимите формати, размери и стойности.

### **Заключение**

Уязвимостите от тип CWE-74 са сериозен риск за сигурността, тъй като позволяват на нападателите да манипулират поведението на приложенията чрез неподходящо обработени входни данни. Правилното неутрализиране и валидиране са ключови за защитата срещу този тип атаки.

The Open Web Application Security Project (OWASP) , е световно призната организация посветена върху подобрението на софтуената сигурност. Тя осигурява безплатни ресурси, инструменти и образователни материали за помощ при разработка , и кибер защита на приложения. OWASP топ 10 е списък от десетте най-често срещани рискове за сигурността на приложенията. Чрез адресиране на тези рискове приложенията създадети се справят по-добре в това да защитават чувствителни данни на потребителите и предпазват приложението от потенциални атаки.

В този реферат ще разгледаме Injection (Инжектиране). През 2017 това е оценено ,като номер едно заплаха за приложенията и пада на номер три през 2021. Инжекционните атаки представляват широк спектър от атаки ,но се характеризират със ситуацията: Нападателя предоставя некоректни входни данни (input) към програмата. Тези данни се обработват от интерпретатор като заявка или команда в резултат, от което програмата си променя поведението. Този тип атаки са едни от най-старите и опасни насочени предимно към уеб приложения. Те могат да доведат до кражба или загуба на данни. Отказ на услуга (micro service) , както и до срив на системата. Основна причина е недостатъчно валидиране на входните данни.

Основната причина този тип атака да е в класацията е това че е популярен и стар т.е добре научен и съществува голям набор от приложения даващи възможност на неопитни нападатели да е извършват с висок процент на успеваемост.

Приложение е уязвимо към този вида атака , ако:

Потребителските данни не се валидират, филтрират или почистват от приложението.

Динамични заявки или не параметризирани извиквания без контекстуално избягване се използват директно в интерпретатора.

Враждебни данни се използват в параметри за търсене в ORM (Object-Relational Mapping), за да се извлекат допълнителни и чувствителни записи.

Враждебни данни се използват директно или се конкатенират. SQL или командата съдържат структурата и злонамерените данни в динамични заявки, команди и съхранени процедури.

Често срещани видове инжекции

SQL

NoSQL

Команди на операционната система (OS Command)

Object Relational Mapping (ORM)

LDAP

Expression Language (EL) или Object Graph Navigation Library (OGNL)

Концепцията е идентична за всички интерпретатори.

Методи за откриване на уязвимости

Прегледът на изходния код е най-добрият метод за откриване на уязвимости към инжекции. Силно се препоръчва автоматизирано тестване на всички параметри, хедъри, URL адреси, бисквитки, JSON, SOAP и XML входни данни. Програмистие могат да включат статично (SAST), динамично (DAST) и интерактивно (IAST) тестване за сигурност на приложенията в CI/CD процеса, за да идентифицират инжекционни слабости преди внедряването в продукционна среда.

Как да се предпазим?

За да предотвратим инжекционни атаки, е важно да разделяме данните от командите и заявките. Това означава, че когато работим с данни, не трябва да ги вграждаме директно в командите за изпълнение.

Най-добрият подход е да използваме безопасен API, който автоматично се грижи за параметризацията на заявките, или да преминем към използване на инструменти за обектно-релационно картографиране (ORM), които се грижат за защитата. Въпреки това, дори когато използваме параметризирани процедури, можем да имаме уязвимости, ако те конкатенират данни и заявки или изпълняват враждебни данни чрез функции като exec().

Друг важен аспект е да правим валидация на входящите данни на сървъра, като се уверим, че те са безопасни за обработка. Това обаче не е пълна защита, тъй като в някои приложения е необходимо използването на специални символи, например в текстови полета или мобилни приложения.

Ако все пак използваме динамични заявки, трябва да избягваме специални символи, като използваме правилния синтаксис за тяхното "избягване" (escaping), съобразен с конкретния интерпретатор. Важно е да се знае, че не можем да избягваме структури като имена на таблици или колони, което прави опасно използването на потребителски вход за тези елементи.

Накрая, за да предотвратим разкритие на твърде много данни при SQL инжекция, е препоръчително да използваме ограничения като “LIMIT” в SQL заявките, които да ограничат броя на върнатите резултати.

### **Видове SQL инжектиране:**

* **Извличане на скрити данни**: Нападателят заобикаля ограниченията и получава достъп до скрити данни, като модифицира SQL заявките.
* **Промяна на логика на приложението**: Нападателят може да манипулира логиката на заявките (например, за да заобиколи логина).
* **UNION атаки**: Нападателят извлича данни от различни таблици в базата данни, използвайки оператора UNION.
* **Сляпо SQL инжектиране**: Когато приложението не връща резултати от заявката, нападателите могат да използват техники за време или логически условия, за да разкрият данни.

Най-добрият начин за предотвратяване на SQL инжектиране е чрез **параметризирани заявки** (подготвени изявления), които предотвратяват съединяването на входни данни от потребителя в SQL заявки.

"Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent

to an interpreter as part of a command or query. The attacker’s hostile data can trick the

interpreter into executing unintended commands or accessing data without proper authorization.;

[1] https://www.owasp.org/images/7/72/OWASP\_Top\_10-2017\_%28en%29.pdf.pdf

1. https://www.acunetix.com/blog/articles/injection-attacks/

2. https://www.acunetix.com/websitesecurity/sql-injection/

3. https://www.acunetix.com/websitesecurity/sql-injection2/

4. https://www.acunetix.com/blog/articles/blind-out-of-band-sql-injection-vulnerability-testing-added-acumonitor/

5. https://portswigger.net/web-security/sql-injection

6. https://www.acunetix.com/blog/web-security-zone/nosql-injections/

7. https://www.netsparker.com/blog/web-security/what-is-nosql-injection/

8. https://www.acunetix.com/blog/web-security-zone/os-command-injection/

9. https://www.synopsys.com/glossary/what-is-ldap-injection.html"