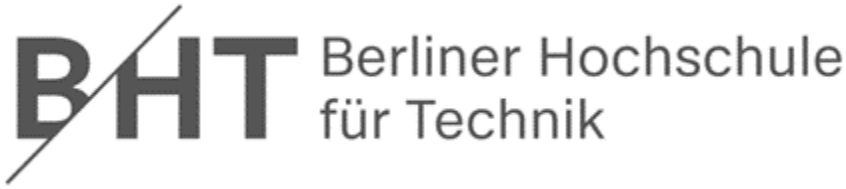
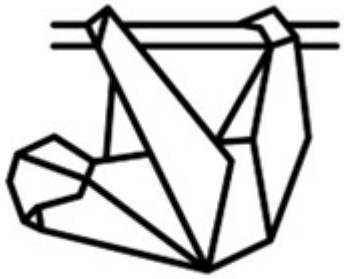


Fachbereich VI - Informatik und Medien
Studiengang IT-Sicherheit Online / Medieninformatik

Vorbereitung 10 - Bedrohungsmodellierung, Programmierung



PARETO CONSULTING

Modul:	Sicherheitsmanagement
Dozent:	Sven Zehl
vorgelegt von:	Gruppe 1 Christine Kuczera Dirk Drutschmann Hicham Naoufal Michael Schröter Jan Zimmermann Ivo Valls

Aufgabe 1a) Recherchieren Sie über die hier aufgelisteten Angriffsvektoren und durchdringen Sie jeden einzelnen.

1. Injection 2) Authentifizierung 3) Broken Authentification: Hash, PW, Verschlüsselung
2. Autorisierung, XSS, CSRF 5) Deserialisierung (Java) 6) Logging

Antwort

1. Injection Der Angriffsvektor Injection bezieht sich auf eine Sicherheitslücke oder Schwachstelle in einer Anwendung oder einem System, die es einem Angreifer ermöglicht, schädlichen Code einzuführen oder unerwünschte Befehle auszuführen. Bei einer Injection-Attacke werden normalerweise unzureichend überprüfte oder validierte Benutzereingaben verwendet, um schädlichen Code einzuschleusen.

SQL-Injection: Bei dieser Art von Angriff werden schädliche SQL-Befehle in eine Anwendung oder eine Datenbank eingeschleust, um vertrauliche Informationen zu stehlen, Daten zu manipulieren oder das System zum Absturz zu bringen.

Command Injection: Hierbei werden schädliche Befehle in eine Systemanwendung oder ein Betriebssystem eingeschleust, um Kontrolle über das betroffene System zu erlangen und unerwünschte Aktionen auszuführen.

Cross-Site Scripting (XSS): Diese Art von Angriff beinhaltet die Einschleusung von schädlichem Skriptcode in Webseiten, die dann von anderen Benutzern ausgeführt wird. Dadurch kann der Angreifer sensible Informationen stehlen oder die betroffenen Benutzer auf betrügerische Seiten umleiten.

LDAP-Injection: LDAP-Injection tritt auf, wenn schädliche Daten in eine LDAP-Anfrage eingeschleust werden, um unberechtigten Zugriff auf das Verzeichnisdienstsystem zu erlangen oder Informationen zu stehlen.

Injection-Angriffe sind potenziell gefährlich und können zu erheblichen Sicherheitsverletzungen führen. Um solche Angriffe zu verhindern, ist es wichtig, Eingaben sorgfältig zu überprüfen, Parameter zu validieren und Sicherheitsmechanismen wie Prepared Statements, Stored Procedures und Input Validation zu verwenden.

2. Authentifizierung

Der Angriffsvektor Authentifizierung bezieht sich auf potenzielle Schwachstellen oder Angriffsmethoden, die im Zusammenhang mit dem Prozess der Identifizierung und Überprüfung der Identität eines Benutzers auftreten können. Authentifizierung bezieht sich auf den Mechanismus, der sicherstellt, dass der Benutzer tatsächlich die behauptete Identität besitzt, um auf bestimmte Ressourcen, Systeme oder Informationen zugreifen zu können.

Brute-Force-Angriff: Bei einem Brute-Force-Angriff versucht ein Angreifer, sich Zugang zu einem Konto oder System zu verschaffen, indem er systematisch alle möglichen Kombinationen von Benutzernamen und Passwörtern ausprobiert, bis er die richtige Kombination findet.

Passwort-Phishing: Bei dieser Art von Angriff versucht ein Angreifer, Benutzer dazu zu bringen, ihre Login-Daten auf gefälschten Websites oder über betrügerische E-Mails preiszugeben, die vorgeben, von vertrauenswürdigen Quellen zu stammen.

Passwort-Wiederverwendung: Dieser Angriff basiert darauf, dass Benutzer dasselbe Passwort für mehrere Konten verwenden. Wenn ein Angreifer Zugriff auf ein Konto erhält, kann er versuchen, dasselbe Passwort für andere Konten des Benutzers zu verwenden.

Session-Hijacking: Hierbei versucht ein Angreifer, eine laufende Sitzung (Session) eines authentifizierten Benutzers zu übernehmen, um Zugriff auf das Konto oder die Ressourcen des Benutzers zu erhalten.

Man-in-the-Middle (MitM)-Angriff: Bei einem MitM-Angriff positioniert sich der Angreifer zwischen dem Benutzer und dem Authentifizierungsserver, um die Kommunikation abzufangen, zu manipulieren oder sensible Informationen wie Benutzernamen und Passwörter zu stehlen.

Um Angriffe auf die Authentifizierung zu verhindern, sollten bewährte Sicherheitspraktiken wie die Verwendung starker Passwörter, die Implementierung von Mehr-Faktor-Authentifizierung (MFA), die regelmäßige Überprüfung und Aktualisierung von Sicherheitsprotokollen sowie die Sensibilisierung der Benutzer für Phishing- und Social-Engineering-Techniken umgesetzt werden.

3. Broken Authentification: Hash, PW, Verschlüsselung

Broken Authentication bezieht sich auf eine Sicherheitslücke oder einen Schwachpunkt im Authentifizierungsmechanismus einer Anwendung oder eines Systems, die es Angreifern ermöglicht, die Identität von Benutzern zu übernehmen, sich ohne Berechtigung Zugang zu Konten zu verschaffen oder vertrauliche Informationen zu stehlen. Diese Schwachstelle kann verschiedene Formen annehmen, einschließlich unsicherer Handhabung von Passwörtern, schwacher Hash-Funktionen oder unzureichender Verschlüsselung.

Passwort-Hashing: Passwort-Hashing bezieht sich auf den Prozess, bei dem ein Passwort in einen nicht umkehrbaren Hash-Wert umgewandelt wird. Ein sicherer Hash-Algorithmus sollte verwendet werden, um sicherzustellen, dass der Hash-Wert nicht in das ursprüngliche Passwort umgerechnet werden kann. Bei Broken Authentication kann dies bedeuten, dass entweder unsichere Hash-Algorithmen oder schlecht gesalzene Hashes verwendet werden, was es Angreifern ermöglicht, den Hash-Wert zu berechnen und das ursprüngliche Passwort abzuleiten.

Schwache Passwörter: Schwache Passwörter sind leicht zu erraten oder zu knacken. Wenn eine Anwendung eine schwache Passwortrichtlinie hat oder es Benutzern erlaubt, unsichere Passwörter zu verwenden, erleichtert dies Angreifern den Zugriff auf Konten durch Brute-Force-Angriffe oder das Ausprobieren häufig verwendeter Passwörter.

Unsichere Übertragung von Authentifizierungsdaten: Wenn Authentifizierungsdaten, wie Benutzername und Passwort, während der Übertragung nicht ausreichend verschlüsselt werden, können Angreifer diese Informationen abfangen und auslesen. Dies kann geschehen, wenn eine Anwendung keine sichere HTTPS-Verbindung verwendet.

Session-Management: Schwachstellen im Session-Management können zu Session-Hijacking führen, bei dem ein Angreifer eine laufende Sitzung übernimmt und sich als authentifizierter Benutzer ausgibt. Dies kann geschehen, wenn Session-IDs unsicher generiert, nicht ausreichend geschützt oder nicht ordnungsgemäß abgelaufen sind.

Um Broken Authentication zu verhindern, sollten bewährte Sicherheitspraktiken wie die Verwendung starker Passwortrichtlinien, sichere Hash-Funktionen und Salzverfahren, die sichere Übertragung von Daten über verschlüsselte Kanäle, die Implementierung von Session-Management-Mechanismen und die regelmäßige Aktualisierung von Sicherheitspatches und -richtlinien umgesetzt werden. Zusätzlich kann die Implementierung von Multi-Faktor-Authentifizierung (MFA) die Sicherheit erhöhen, indem mehrere Authentifizierungsfaktoren erforderlich sind, um Zugriff auf ein Konto zu erhalten.

4. Autorisierung, XSS, CSRF

Autorisierung: Autorisierung bezieht sich auf den Prozess der Überprüfung und Gewährung von Berechtigungen für einen authentifizierten Benutzer, um auf bestimmte Ressourcen, Funktionen oder Aktionen zuzugreifen. Ein Angriffsvektor im Bereich der Autorisierung bezieht sich auf Schwachstellen oder Angriffsmethoden, die es einem Angreifer ermöglichen, unerlaubten Zugriff auf geschützte Ressourcen oder Aktionen zu erlangen. Dies kann beispielsweise durch Umgehen von Berechtigungsprüfungen, Ausnutzen von fehlerhaften Berechtigungseinstellungen oder Manipulation von Zugriffsrechten erreicht werden.

XSS (Cross-Site Scripting): XSS ist eine Art von Sicherheitslücke, bei der ein Angreifer schädlichen Code (meistens JavaScript) in eine vertrauenswürdige Webseite einschleust. Wenn ein Benutzer diese infizierte Seite besucht, wird der eingeschleuste Code im Browser des Benutzers ausgeführt. Dies ermöglicht dem Angreifer, Benutzerdaten zu stehlen, Sitzungen zu übernehmen, schädliche Aktionen im Namen des Benutzers auszuführen oder die Webseite zu manipulieren. XSS-Angriffe können in verschiedenen Formen auftreten, einschließlich gespeicherter XSS und reflektierter XSS.

CSRF (Cross-Site Request Forgery): CSRF ist eine Art von Angriff, bei dem ein Angreifer einen Benutzer dazu bringt, ungewollte Aktionen auf einer Webseite auszuführen, ohne dass der Benutzer es bemerkt. Der Angreifer täuscht den Benutzer, indem er eine bösartige Anfrage sendet, die im Namen des Benutzers abgelegt wird, wenn dieser bereits in der Webseite authentifiziert ist. Dadurch kann der Angreifer schädliche Aktionen durchführen, die vom Benutzer autorisiert sind, wie z.B. das Ändern von Passwörtern, Durchführen von Transaktionen oder Löschen von Daten.

Um diese Angriffsvektoren zu verhindern, sollten entsprechende Sicherheitsmaßnahmen ergriffen werden:

Autorisierung: Implementierung einer soliden und gründlichen Berechtigungsprüfung, um sicherzustellen, dass Benutzer nur auf die für sie autorisierten Ressourcen zugreifen können. Überprüfen der Berechtigungen auf Serverseite und nicht allein auf Clientseite.

XSS: Eingabevalidierung und -bereinigung auf allen Ebenen, insbesondere bei der Darstellung von Benutzereingaben. Verwendung von Content Security Policy (CSP) und Escape/Encoding-Funktionen, um potenziell schädlichen Code zu neutralisieren.

CSRF: Implementierung von Anti-CSRF-Maßnahmen wie der Verwendung von CSRF-Token, um sicherzustellen, dass nur legitime Anfragen akzeptiert werden. Validierung der Herkunft (Origin) von Anfragen, um sicherzustellen, dass sie von der erwarteten Quelle stammen.

5. Deserialisierung (Java)

Bei der Deserialisierung handelt es sich um den Prozess des Konvertierens eines serialisierten Objekts, das in eine Datenstruktur oder einen Datenstrom geschrieben wurde, zurück in ein Objekt. Dieser Prozess wird in vielen Programmiersprachen verwendet, einschließlich Java.

Ein Angriffsvektor der Deserialisierung tritt auf, wenn ein Angreifer die Deserialisierungsfunktion ausnutzt, um schädlichen Code einzuschleusen und auszuführen. Dies kann geschehen, wenn ein unsicherer Deserialisierungsmechanismus verwendet wird, der es einem Angreifer ermöglicht, bösartigen Code in den Deserialisierungsprozess einzuschleusen. Der Angreifer kann manipulierte oder böswillig erstellte Serialisierungsdaten bereitstellen, die von der Anwendung deserialisiert werden.

In Java kann die Deserialisierung von Objekten potenziell gefährlich sein, da sie zu schwerwiegenden Sicherheitslücken führen kann. Wenn eine Anwendung unsichere Deserialisierungsmechanismen verwendet oder nicht ausreichend überprüft, ob die deserialisierten Daten sicher sind, kann ein Angreifer Code einschleusen und ausführen, der die Anwendung kompromittiert.

Um Angriffsvektoren der Deserialisierung zu verhindern, ist es wichtig, sichere Deserialisierungstechniken zu verwenden. Dazu gehören das Validieren und Überprüfen der deserialisierten Daten, das Verwenden von Whitelists anstelle von Blacklists, um zulässige Klassen zu definieren, das Verhindern des Deserialisierens von vertraulichen oder sensiblen Daten und das Aktualisieren der verwendeten Deserialisierungsbibliotheken, um von Sicherheitspatches zu profitieren. Es ist auch empfehlenswert, Deserialisierungsoperationen auf vertrauenswürdige und privilegierte Teile der Anwendung zu beschränken.

6. Logging

Der Angriffsvektor Logging bezieht sich auf eine potenzielle Schwachstelle oder eine Möglichkeit für Angreifer, das Logging-System einer Anwendung auszunutzen, um unerwünschte Ergebnisse zu erzielen oder sensible Informationen zu erlangen. Logging ist der Prozess des Aufzeichnens von Ereignissen, Aktivitäten oder Fehlern in einer Anwendung oder einem System, um sie zu überwachen, zu analysieren oder zur Fehlerbehebung zu verwenden.

Ein Angreifer kann verschiedene Techniken nutzen, um das Logging-System zu manipulieren oder auszunutzen:

Falsche Informationen protokollieren: Ein Angreifer kann versuchen, gefälschte oder irreführende Protokollmeldungen zu generieren, um den Betrieb der Anwendung zu stören, den Systemadministrator in die Irre zu führen oder die forensische Analyse zu erschweren.

Sensible Informationen protokollieren: Wenn das Logging-System nicht angemessen geschützt ist, kann ein Angreifer versuchen, vertrauliche Informationen wie Passwörter, Zugangsdaten, Benutzereingaben oder andere sensible Daten in Protokolldateien zu erfassen. Dies könnte zu Datenschutzverletzungen oder Sicherheitsproblemen führen.

Den Log-Verlauf ändern: Ein Angreifer kann versuchen, den Log-Verlauf zu manipulieren, indem er Protokolldateien löscht, verändert oder neue Einträge einfügt. Dadurch könnte er seine eigenen Aktivitäten verbergen, die forensische Analyse erschweren oder die Nachverfolgung von Angriffen und Sicherheitsvorfällen behindern.

Den Log-Mechanismus überlasten: Durch das Erzeugen einer großen Anzahl von Protokolleinträgen oder durch gezielte Angriffe auf das Logging-System kann ein Angreifer versuchen, die Ressourcen der Anwendung zu überlasten oder die Stabilität des Systems zu beeinträchtigen.

Durch die folgenden Sicherheitsvorkehrungen kann das Risiko von Angriffen auf das Logging-System reduziert werden:

- Implementieren von Zugriffsbeschränkungen für das Logging-System, um unbefugten Zugriff zu verhindern.
- Vermeiden der Protokollierung sensibler Informationen wie Passwörter oder Zugangsdaten.
- Überprüfung und Filterung von Benutzereingaben, um das Einfügen schädlicher Daten in Protokolldateien zu verhindern.
- Überwachen und Analysieren der Protokolldateien auf verdächtige Aktivitäten oder Angriffsversuche.
- Implementieren von Sicherheitsmaßnahmen, um die Integrität und Vertraulichkeit der Protokolldateien zu gewährleisten, z.B. durch Verschlüsselung oder digitale Signaturen.
- Regelmäßiges Aktualisieren des Logging-Frameworks oder der Bibliotheken, um von Sicherheitspatches und -aktualisierungen zu profitieren.

Aufgabe 1b) Betrachten Sie nun für Ihren Use Case Asset und Angreifer. Welcher der gerade recherchierten Angriffsvektoren käme diesem Szenario am nächs- ten? Erklären Sie ihn detailliert.

Antwort

Wie auch in Vertiefung 7 durch das Microsoft Threat Modeling Tool festgestellt wurde, ist von den aufgelisteten Angriffsvektoren die SQL Injection für unseren Use Case - ein Webauktionshaus - ein realistischter niederschelliger Angriffsvektor. Ein möglicher Angreifer könnte ein Konkurent sein, welcher aus unserer Datenbank die Kontaktdaten hochkarätiger Kunden auslesen möchte um diesen selbst Angebote schicken zu können.