

Technology
Arts Sciences
TH Köln

A secure web application template

Module:

Systementwurfpraktikum (SYP)
&
Präsentation & Kommunikation (PuK)



Wintersemester 22/23

Einführung

Team 09

Secure Web App

Systemarchitektur

Live Demo

Nächste Schritte

Q&A

Cyber-Angriffe

Finanzdaten

Sensible Daten

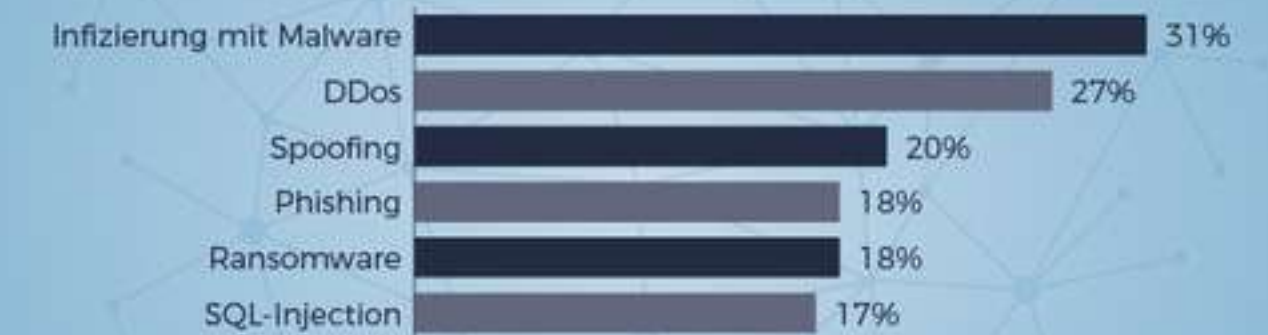
Service ausschalten

Schadsoftware

Malware Verbreitung



DIE HÄUFIGSTEN ANGRIFFSARTEN:







2020 waren 86% der Unternehmen von Cyber-Angriffen betroffen!

Web-Sicherheit

Verschlüsselung der Nutzdaten

Sichere Verbindung

Zugriffskontrolle

Einführung	<div><div><u>Teammitglieder</u></div><div><div><div>Ahmad Al Housseini</div><div>Alpar Gür</div><div>Fabian Ullmann</div><div>Leonel Nguimatsia Tsobguim</div></div><div><div><u>Betreuer</u></div><div>Prof. Dr. Heiko Knospe M. Sc. Andreas Schwenk</div></div></div></div>			
Team 09				
Secure Web App				
Systemarchitektur				
Live Demo				
Nächste Schritte				
Q&A				

Einführung

Team 09

Secure Web App

Systemarchitektur

Live Demo

Nächste Schritte

Q&A

With it becoming increasingly easy for cybercriminals to guess passwords, 2FA is more important than ever. It doesn't seem like a hassle to add an extra step to your login process, but without it you could be leaving your personal information open to cybercriminals who want to steal your personal information, or hack into your bank accounts, or even take over your website.

2FA-Authentication

Adding the extra step to account access means a user has to struggle to access your personal information. If you add a knowledge factor to your bank account, a cybercriminal who knows your password won't be able to access the account without having your phone when it receives the verification code.

Developers frequently underestimate the difficulty of implementing a reliable access control mechanism. Many of these schemes were not deliberately designed, but have simply evolved along with the web site. In these cases, access control rules are inserted in various locations all over the code. As the site nears deployment, the rules become so unwieldy that it is almost impossible to understand. Many of these flawed access control schemes are not difficult to discover and exploit. Frequently, all that is required is to craft a request for functions or content that should not be available to the user.

Broken Authentication is the only category not to have any Common Vulnerability and Exposure (CVE) mapped to the included CWEs, so a default exploit and impact weights of 5.0 are factored into their scores.

One specific type of access control problem is administrative interfaces that allow site administrators to manage a site over the Internet. Such features are frequently used to allow site administrators to efficiently manage users, data, and content on their site. Administrative interfaces support a variety of administrative roles to allow finer-grained site management. Due to their power, these interfaces are frequently prime targets for attack by both hackers and insiders.

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious code is injected into the trusted web page. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side-script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the browser will execute it. This can result in session tokens, or other sensitive information being hijacked, and used with that site. These scripts can even rewrite the content of the page and used for phishing.

The primary form of SQL injection consists of inserting a malicious query or the insertion of code into user-input variables. However, failures in concatenating with SQL commands and escaping special characters can lead to a direct attack injecting malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a query, the dynamic SQL becomes malicious.

The injected code is then executed by the database, inserting a text string and appending a new command. Because the inserted command may have additional strings appended to it before it is executed, the malefactor terminates the injected string with a comment mark "--". Subsequent text is ignored at execution time.

Since applications can communicate either with or without TLS (or SSL), it is necessary for the client to request that the server set up a TLS connection.[2] One of the main ways of achieving this is to use a different port number for TLS connections. Port 80 is typically used for HTTP traffic while port 443 is the common port used for HTTPS traffic.

TLS-Connection

Another mechanism to make a protocol-specific STARTTLS connection to TLS is to use the mail and news protocols. For example, when using the mail and news protocols, the client and server have agreed to use TLS, they negotiate a handshake. The protocols use a handshake with an asymmetric cipher to establish not only cipher settings but also a session-specific key.

Exposure, whether it was a root cause or a contributing factor, was the most common failure type in the 2021 OWASP Top 10. The renewed focus here is on failures to sensitive data exposure or system compromise.

OWASP Cross-site Scripting is now part of this category in this edition. A04:2021-Insecure Design is a new category for 2021, with a focus on risks related to design flaws. If we genuinely want to “move left” as an industry, it calls for more use of threat modeling, secure design patterns and principles, and reference architectures.

Misconfiguration moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category.

A06:2021-Vulnerable and Outdated Components was previously titled Using Components with known vulnerabilities and is #2 in the Top 10 community survey, but also had enough data to make the top 10. This category moves up from #9 in 2017 and is a known issue that we struggle to test for.

Deserialization Failures was previously Broken Authentication and Authentication Failures. It is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.

Software and Data Integrity Failures is a new category for 2021, focusing on integrity assumptions related to software updates, critical data, and CI/CD pipelines with no surveying strategy.

CVE/CVSS (Common Vulnerability and Exposure) data mapped to the top 10 CVEs in this category. Deserialization from 2017 is now a part of this larger category.

A09:2021-Security Logones and Monitoring Failures was previously Broken Authentication and Authentication Failures. It is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute an unwanted action on a trusted site. The user is tricked into executing the action by the attacker's choosing. If the victim is a normal user, the attacker might trick the user into performing state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. The values are usually used to index a fixed-size table called a hash table. Use of a hash function to index a hash table is called hashing or scatter storage addressing.

Password Hashing

Hash functions are used in data storage and retrieval applications to access data in a small and nearly constant amount of storage space only. They require an amount of storage space only fractionally greater than the total space required for the data or records themselves. Hashing is a computationally and storage space-efficient form of data access that avoids the non-constant access time of ordered structures, and the often exponential growth of state spaces of large or variable-length keys.

94% of applications were tested for password hashing.

JWT

The Web Authentication API (also known as WebAuthn) is a specification written by the W3C and FIDO with the participation of Google, Mozilla, Microsoft, Yubico, and others. The API allows servers to register and authenticate users using public key cryptography instead of a password. It allows the strong authentication now built into devices, like Windows Hello or Apple's Touch ID. Instead of a password, a private-public keypair (known as a credential ID) is created for a website. The private key is stored securely on the user's device; a public key generated credential ID is sent to the server.

WebAuthn

WebAuthn is a specification written by the W3C and FIDO with the participation of Google, Mozilla, Microsoft, Yubico, and others. The API allows servers to register and authenticate users using public key cryptography instead of a password. It allows the strong authentication now built into devices, like Windows Hello or Apple's Touch ID. Instead of a password, a private-public keypair (known as a credential ID) is created for a website. The private key is stored securely on the user's device; a public key generated credential ID is sent to the server.

The server can then use that key to prove the user's identity. The data shows that the use of one original server.[5] The data shows that the use of one original server.[5]

Monitoring and Logging is expanded to include the use of one original server.[5] The data shows that the use of one original server.[5]

Reverse Proxy is expanded to include the use of one original server.[5] The data shows that the use of one original server.[5]

Policy default-src 'none'; script-src 'self'; style-src 'self'; font-src 'self'; connect-src 'self'; manifest-src 'self'; worker-src 'none'; base-uri 'self'; form-action 'self';

Einführung

Team 09

Secure Web App

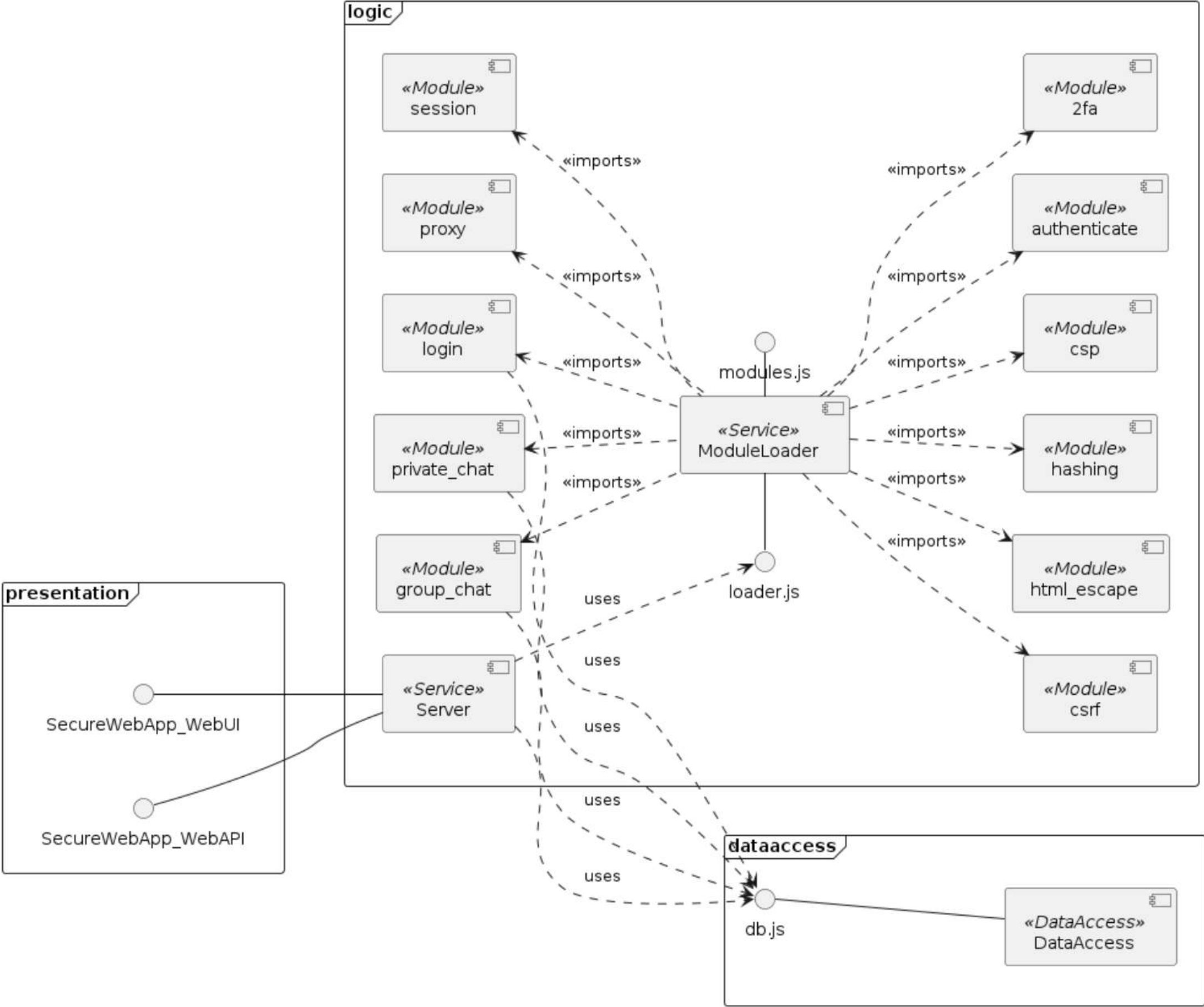
Systemarchitektur

Live Demo

Nächste Schritte

Q&A

Komponenten



Einführung	<div><h2>Deployment</h2><pre>graph TD; Client["«client» Browser"] -- https --> AppServer["«device» ApplicationServer"]; subgraph AppServer ["«device» ApplicationServer"]; direction TB; Proxy["«proxy» ReverseProxy"]; subgraph WebServer ["«web server» Apache"]; end; Proxy -- http --> Node["«execution environment» Node"]; subgraph NodeEnv ["«execution environment» Node"]; direction TB; SecureWebApp["SecureWebApp"]; end; Node -- mysql --> DBServer["«db server» MySQLServer"]; subgraph DBServer ["«db server» MySQLServer"]; direction TB; MySQLStorage["MySQLStorage"]; subgraph MySQLStorageEnv ["MySQLStorage"]; direction TB; SecureWebApp_DB["SecureWebApp_DB.sql"]; end; end; end; end;</pre></div>
Team 09	
Secure Web App	
Systemarchitektur	
Live Demo	
Nächste Schritte	
Q&A	

Einführung
Team 09
Secure Web App
Systemarchitektur
Live Demo
Nächste Schritte
Q&A



Einführung	Aktueller Stand	
Team 09	<div><div>Kernsystem<ul style="list-style-type: none">Hauptkomponenten ✓Laufzeitumgebung & Datenbanken ✓Apache Proxy ○Installationsskript / Docker Container ○</div><div><div>Funktionen<ul style="list-style-type: none">Registrierung ✓Einloggen ✓2-Faktor Authentifizierung ○Benutzereinstellungen ○AdminfunktionalitätenChatfunktionalität</div><div><div>Sicherheit<ul style="list-style-type: none">Content-Security-Policy ✓Password Hashing ✓Authentifikation ✓Schutz gegen XSS ✓Schutz gegen SQL-Injection ✓Logging ○TLS-Verbindung ○Schutz gegen CSRF</div></div></div></div>	
Secure Web App		
Systemarchitektur		
Live Demo		
Nächste Schritte		
Q&A		

Einführung
Team 09
Secure Web App
Systemarchitektur
Live Demo
Nächste Schritte
Q&A

