

SECURITY

MOTIVATION

UM WAS GEHT ES?

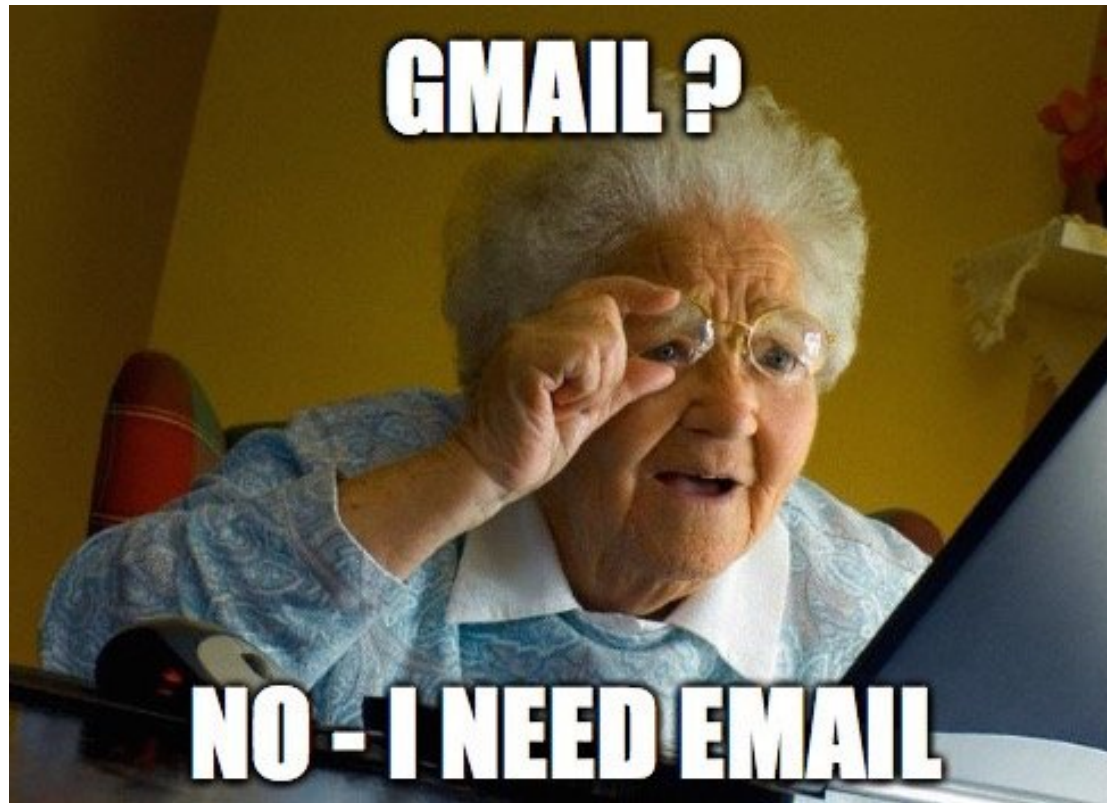
- Datendiebstahl
- Datenveränderung
- Computersabotage
- Computerbetrug

Speaker notes

- Computersabotage und Computerbetrug bauen natürlich auf Datendiebstahl und Veränderung auf

INTERNETNUTZUNG

- (fast) jeder nutzt das Internet



Speaker notes

- Ja, auch technisch nicht versierte Leute nutzen das Internet
- Daher können wir Security nicht auf den Nutzer abwälzen
- Web Anwendungen müssen sicher gebaut sein, um Nutzer vor Gefahren zu schützen

INTERNETNUTZUNG

- Das Internet wird für (fast) alles verwendet
- Zum Beispiel:
 - Soziale Netzwerke
 - Shopping
 - Banking

Speaker notes

- Besonders seit Corona läuft noch mehr online ab.
- Grade diese drei Beispiele sind besonders sicherheitskritisch.

INTERNETNUTZUNG

- Oft geht es um sicherheitskritische Daten
- Zum Beispiel:
 - Privatsphäre & Datenschutz
 - Geld Transaktionen
 - Identitätsdiebstahl

Speaker notes

- Die meisten Datenleaks veröffentlichen private Daten. Es geht weniger um finanziellen Schaden, sondern häufig um Privatsphäre und Datenschutz
- Finanzieller Schaden ist bei veröffentlichten Daten nicht wirklich "messbar".
- Identitätsdiebstahl: Cryptomining unter Elon Musk posten

WEBANWENDUNGEN

- Wind von überall erreichbar
 - Nicht wie ein Bankautomat
- IT-Sicherheit ist keine Einstiegshürde
 - Jeder kann mit einem Tutorial einen Webshop schreiben

Speaker notes

- Ein Angreifer kann bei einer Webanwendung überall auf der Welt sitzen
- Angreifer gehen in der Masse der Anfragen unter
- Service ist 24/7 verfügbar

UNSICHERE WEBANWENDUNGEN

- Selbst die Größten sind nicht sicher
- Man hört immer wieder von Datenleaks oder Einbrüchen
 - Twitch
 - Facebook
 - etc.
- Größere Webanwendungen sind attraktivere Ziele

Speaker notes

- Größere Firmen landen mit ihren Datenleaks auch eher in den Medien

ATTRAKTIVE ZIELE

- Einige Ziele sind sehr attraktiv
 - Twitch
 - Facebook
 - Banken
 - Passwortmanager
- Dies liegt
 - An der Größe
 - An den verwalteten Daten
- Nicht lukrative Ziele müssen trotzdem sicher sein!

Speaker notes

- Natürlich sind einige Ziele lukrativer als andere
- Trotzdem müssen alle Anwendungen über ein Mindestmaß an Sicherheit verfügen

NACHTEILE VON SECURITY

- Security bringt Nachteile mit sich
- Konkurriert mit der Benutzbarkeit
 - Zwei Faktor Authentifizierung (2FA)
- Höhere Entwicklungskosten
- Komplexere Architektur
- Höherer Ressourcenverbrauch

WIE VIEL SICHERHEIT BRAUCHE ICH?

- "it depends"
- Es kommt an auf
 - Die Anforderungen
 - Eas Budget
 - Die Domäne
 - Rechtliche Rahmenbedingungen
- Security ist eine Qualitätsanforderung

Speaker notes

- In einigen Bereichen wird Security direkt vom Kunden/Auftraggeber/Arbeitgeber angefordert
- Z.B. bei Banken ist Sicherheit sehr wichtig.
- Im Endeffekt ist Security eine Qualitätsanforderung.
- Jeder muss für seine Anwendung in einem gewissen Rahmen festlegen wie viel Security man braucht.

MINDESTMASS AN SECURITY

- Nicht immer die wichtigste Qualitätsanforderung
- Ein Mindestmaß muss vorhanden sein
- Dieses Mindestmaß schauen wir uns nun an

ANGRIFFSMETHODEN

ANGRIFFSMETHODEN

- Request-Manipulation
- Directory Traversal
- SQL-Injection
- Session Hijacking
- Cross-Site-Scripting
- Cross-Site-Request-Forgery
- Man-In-The-Browser
- Phishing
- Denail-Of-Service

REQUEST-MANIPULATION

Wer hat davon mitbekommen?



Speaker notes

- <https://lilithwittmann.medium.com/wenn-die-cdu-ihren-wahlkampf-digitalisiert-a3e9a0398b4d>
- Die Sicherheitslücke bei CDU Connect war durch einfache Request Manipulation zu finden.
- Wie Lilith Wittmann in ihrem Beitrag dazu schreibt, konnte sie die Daten der Anwendung einfach über die Backendschnittstellen abfragen.

REQUEST-MANIPULATION

- *"www.some-domain.de/users/41"*
- Gibt es vielleicht auch einen user 42?
- Alle öffentlichen Schnittstellen können aufgerufen werden

Speaker notes

- Security through obscurity ist keine ordentliche Security
- Nur weil jemand nicht darauf klicken kann, heißt es nicht, dass man es nicht aufrufen kann.

REQUEST-MANIPULATION - LÖSUNG

- Datenzugriff nur für berechtigte und authentifizierte Nutzer
- Evtl. zusätzlich keine monoton aufsteigende Id's

Speaker notes

- Wie wir anhand des CDU Beispiels sehen, ist Request Manipulation leider immer noch ein Thema
- Man kann auch aus den kleinsten Daten wichtige Informationen extrahieren.

DIRECTORY TRAVERSAL

- Ähnlich wie Request-Manipulation
- *"http://www.example.com/index.foo?item=datei1.html"*
- *"http://www.example.com/index.foo?item=../../../Config.sys"*

DIRECTORY TRAVERSAL - LÖSUNG

- Keine sensiblen Daten an öffentlichen Orten ablegen
- Zugriffsrechte auf Ordner absichern
- Pfade als Eingabe müssen überprüft werden

Speaker notes

- Am besten einen eigenen Server für seine öffentlich zugänglichen Daten
- Ansonsten kann man die einzelnen Ordner gegen Zugriffe von außen schützen

SQL-INJECTION



SQL-INJECTION

```
1 var username = "foo@mail.com"; --"  
2 var password = "lala"
```

```
1 var sql = "SELECT * FROM user " +  
2           "WHERE username='" + username + "' " +  
3           "AND password= '" + password + "';";
```

```
1 SELECT * FROM user  
2   WHERE username='foo@mail.com';  
3   --' AND password='lala';
```

Speaker notes

- Wir fügen zusätzlich zum Nutzernamen weiteren SQL Code in den Query ein
- Der Rest des Queries wird damit auskommentiert
- Nutzername muss in diesem Fall zumindest valide sein

SQL-INJECTION - VARIANTEN

```
1 var username = "lala"; DROP TABLE user;--"  
2 var password = "lala"
```

```
1 var username = "lala"; UPDATE password='password' " +  
2           "WHERE username='foo@mail.com';--"  
3 var password = "lala"
```

SQL-INJECTION - VORGEHEN

- Ausprobieren der gängigsten Namen für
 - Datenbanken
 - Tabellen
 - Spalten
- Fehlermeldungen liefern wichtige Informationen

Speaker notes

- Hier sind Fehlermeldungen der Datenbank gemeint, die an eine öffentliche Schnittstelle weitergeleitet werden

SQL-INJECTION - LÖSUNG

- Silver Bullet: Prepared Statements

```
1 var sql = "SELECT * FROM user " +  
2           "WHERE username=:username" +  
3           "AND password=:password";  
4  
5 em.createNativeQuery(sql)  
6     .setParameter("username", username)  
7     .setParameter("password", password)  
8     .getSingleResult();
```

SESSION HIJACKING

- Klauen der Session eines Nutzers
- Raten der Session ID
- Ausspähen der Session ID
- Aussperren des Nutzers durch Passwortänderung

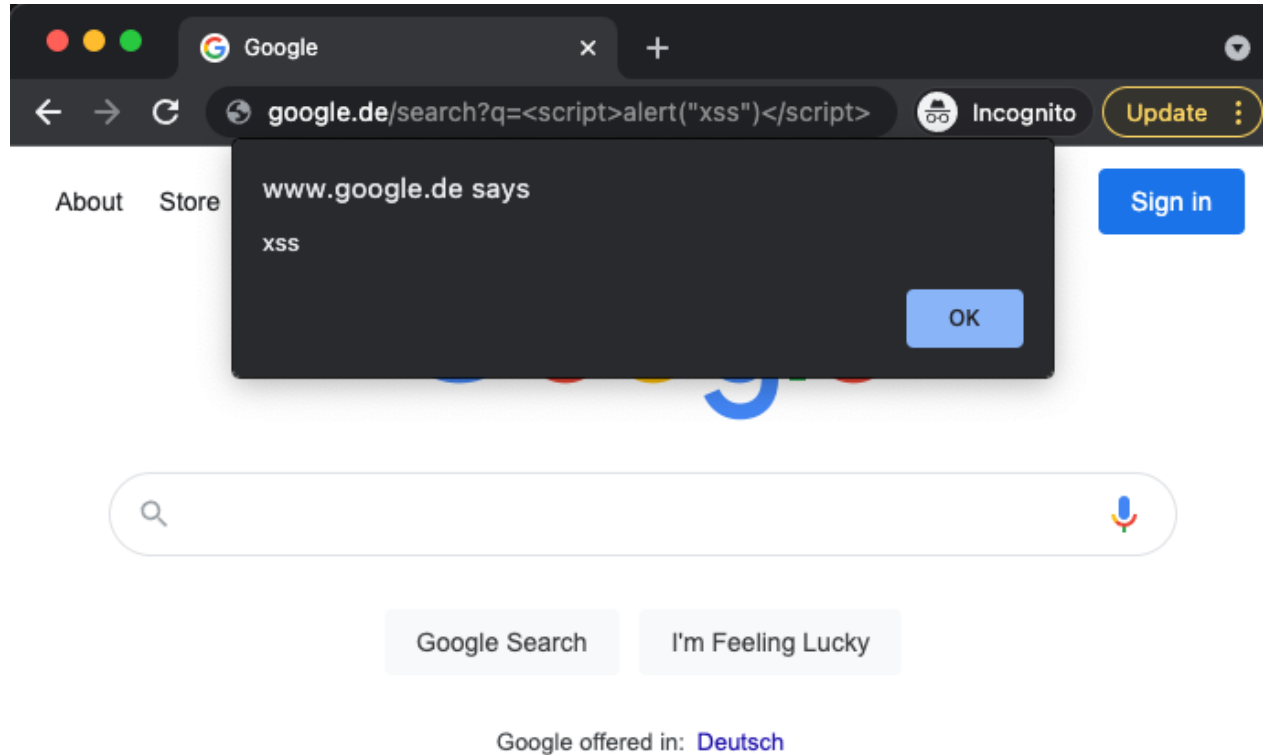
Speaker notes

- Die Session ID kann z.B. durch Cross Site Scripting geklaut werden

SESSION HIJACKING - LÖSUNG

- Binden der Session ID an die IP-Adresse oder Browser
- Größere Session ID wählen
- Passwort ändern verhindern
 - Altes Passwort erneut eingeben

CROSS-SITE-SCRIPTING



Speaker notes

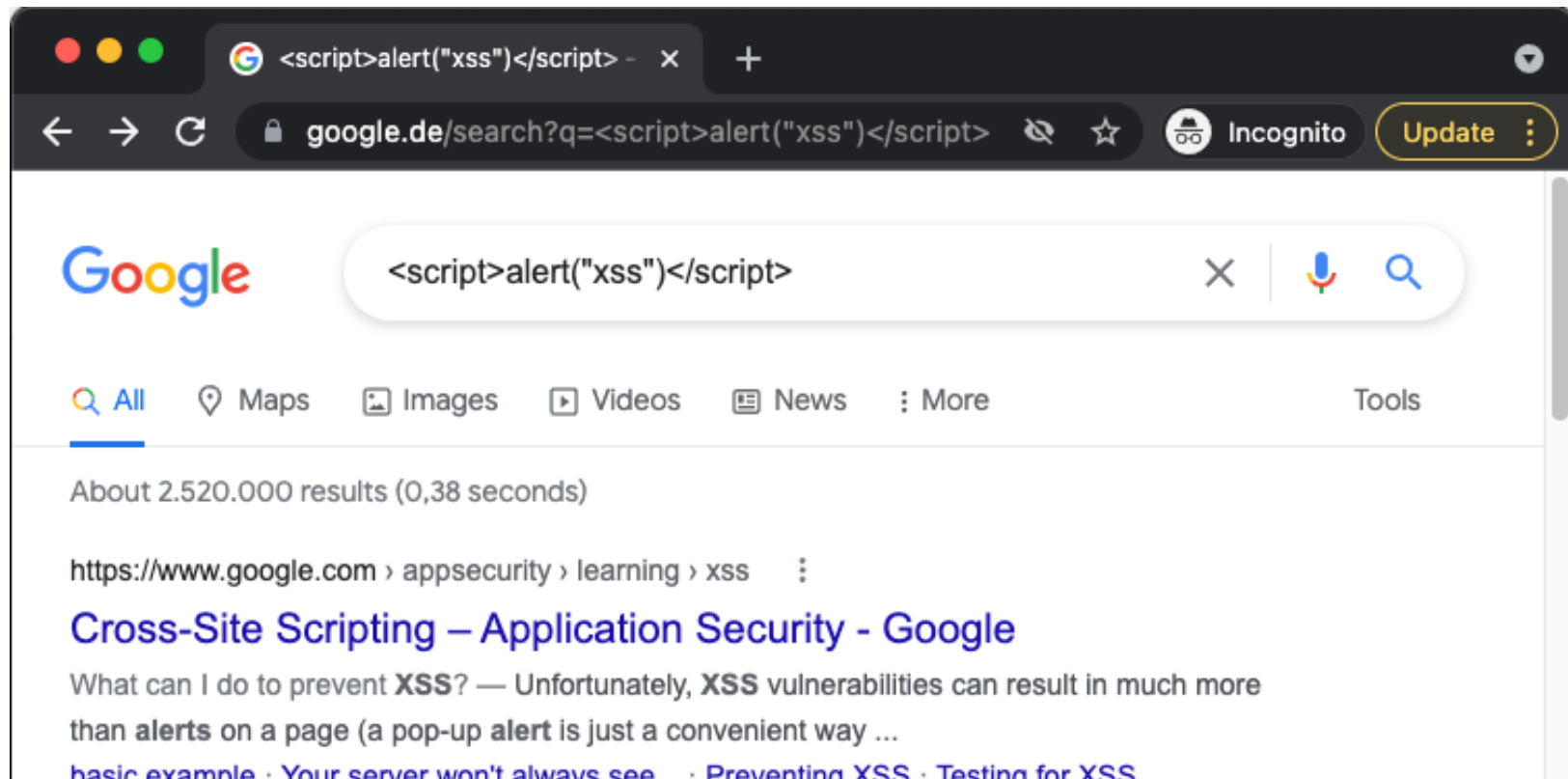
- Über einen Link kann man den Code auch in den Browser eines anderen bringen, dies nennt man nicht-persistentes xss.
- Es gibt auch persistentes xss, z.B. über eine Datenbank (Kommentarfunktion)
 - Eingaben eines Nutzers werden bei anderen Nutzern wieder ausgegeben

CROSS-SITE-SCRIPTING

- Einfügen von JavaScript Code in Webseiten
- Ausführung des Codes durch den Browser
- Möglicher Schaden
 - Verwirrung des Nutzers
 - Weiterleitung auf andere Webseiten
 - Auslesen und Wegschicken von Daten

CROSS-SITE-SCRIPTING - LÖSUNG

- Encoding von Inhalten, die angezeigt werden
- Angular bringt das von Haus aus mit



Speaker notes

- Nicht nur bei der Ausgabe von Inhalten muss geprüft werden. Auch wenn Nutzer Daten an den Server schicken, müssen die Inhalte geprüft werden.
- Dann kann es gar nicht passieren, dass XSS aus dem Backend an den User ausgeliefert wird.

CROSS-SITE-SCRIPTING - LÖSUNG

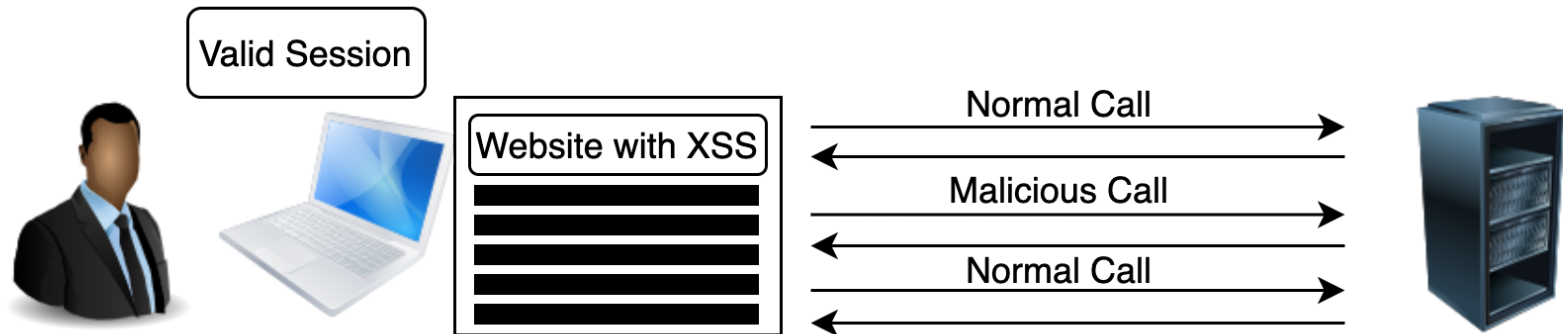
- Content Security Policy
 - Schränkt das Abrufen von Scripts ein
 - Schränkt den JavaScript Code ein, der ausgeführt wird
 - `eval("some-code")`

Speaker notes

- Eine Content Security Policy sagt dem Browser, was er ausführen und laden darf
- Natürlich kommt es auch darauf an, dass der Browser sich richtig verhält

CROSS-SITE-REQUEST-FORGERY

- Mischung aus XSS und Session Hijacking
- Ausnutzung der Session durch XSS
- Script löst im Hintergrund Transaktionen aus



Speaker notes

- Streng genommen ist die Nutzung einer Session Id eine Variante von CSRF. Dies nennt man Session-Riding.
- Normales CSRF beschränkt sich auf das Ausführen von Transaktionen

CROSS-SITE-REQUEST-FORGERY - LÖSUNG

- Siehe XSS
- Webanwendung und Server teilen ein Secret
 - Secret wird bei jedem Request mitgeschickt
 - Secret kann im Cookie oder im Header liegen

Speaker notes

- CSRF baut auf XSS auf. Daher am besten XSS verhindern
- Angular bietet ein `HttpClientXsrfModule` an. Dies kümmert sich um das Teilen eines Secrets (clientseitig)

MAN-IN-THE-MIDDLE (MAN-IN-THE-BROWSER)

- Angreifer schaltet sich zwischen Nutzer und Betreiber
- Zum Beispiel durch einen Fake Webauftritt
- Bei Man-In-The-Browser wird kein Fake Webauftritt gebraucht
 - Der Angreifer manipuliert hier direkt den Browser

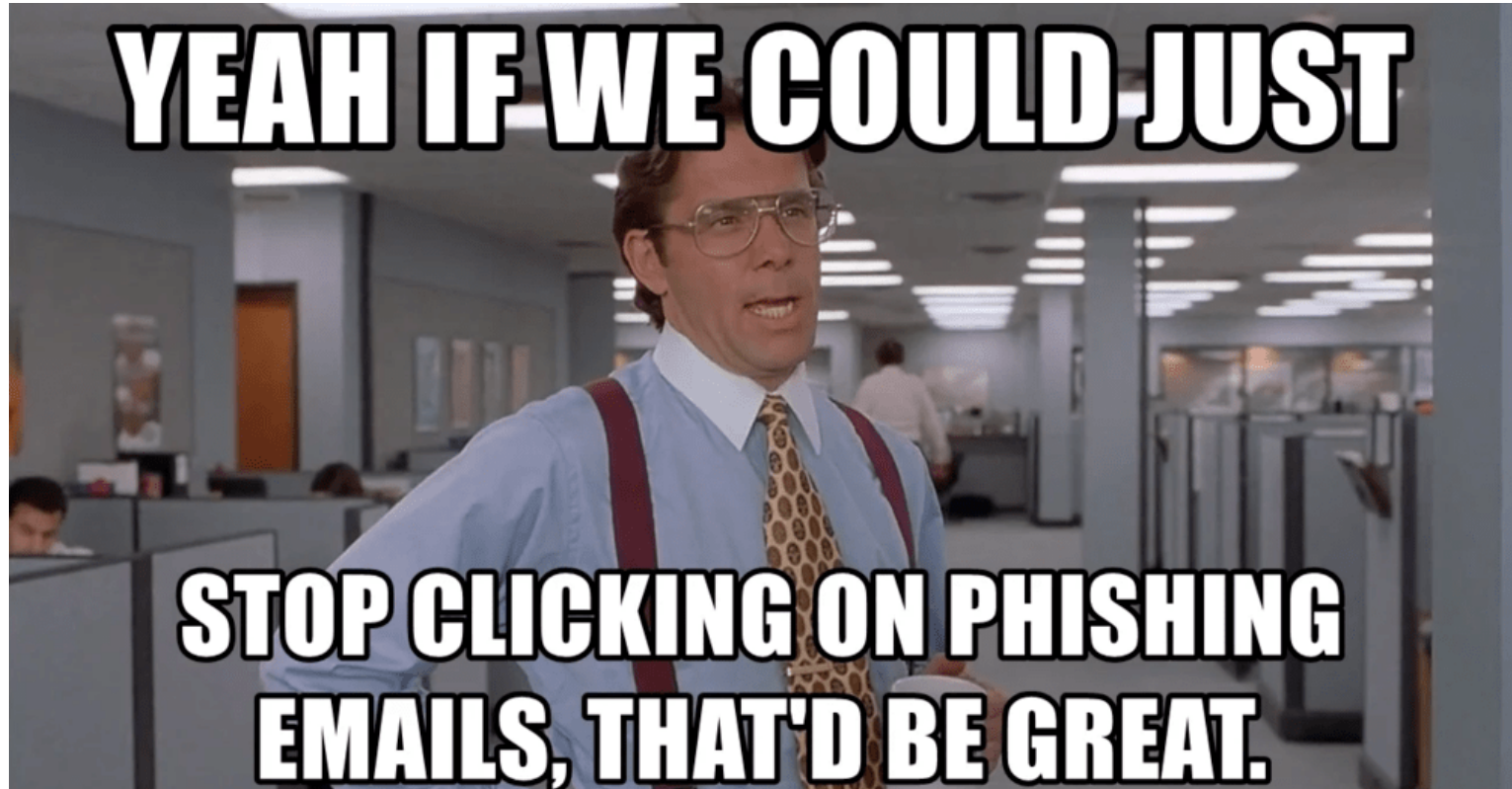
MAN-IN-THE-MIDDLE - LÖSUNG

- HTTPS verwenden (ordentliche Verschlüsselung)
- Zwei Faktor Authentifizierung (2FA)

Speaker notes

- Es gibt natürlich noch weitere Maßnahmen, um das Mitlesen von Daten zu verhindern
- Dies sind die beiden primären Methoden, die wir (als Betreiber) in der Hand haben
- Wenn der zweite Faktor ein Schlüssel ist, kann dies zu einer sehr guten Verschlüsselung der Daten genutzt werden

PHISHING



PHISHING

- Erlangen von persönlichen Daten durchs Vertrauenserschleichung
- Angreifer gibt sich z.B. als Webseiten Betreiber aus
- Basiert auf Social Engineering

PHISHING - LÖSUNG

- 2FA hilft gegen Identitätsdiebstahl
- Nutzer müssen leider mitdenken

Speaker notes

- Mit 2FA muss der Angreifer auch den zweiten Faktor, das Handy, stehlen, um die Identität des Nutzers zu klauen
- Sich im Internet zu bewegen wird immer Mitdenken erfordern

DENAIL-OF-SERVICE

- Überlastung eines Systems durch viele Anfragen
- Im speziellen Fall auch Distributed-Denail-of-Service
 - Hierbei werden Anfragen von vielen Rechnern gestellt
 - Oft durch Botnetze

Speaker notes

- Gehackte Rechner werden zu einem Botnetz zusammengesteckt

DENAIL-OF-SERVICE - LÖSUNG

- Wird im Idealfall vom Server Provider verhindert
- Muster der Angriffe erkennen und auf diese nicht reagieren

ABWEHRMASSNAHMEN

DATA MINING

- Aus Daten auf neue Daten schließen
- So ergeben sich aus wenig Daten viele Informationen
- Empfehlung: Daniel Kriesel "Spiegel Mining"

Speaker notes

- Beispiel: Unterschied zwischen nicht vorhanden (404) und verboten (403) bei fortlaufenden Id's lässt auf die Menge schließen
- Spiegel Mining: <https://www.youtube.com/watch?v=-YpwsdRKt8Q>

ALLGEMEINE ABWEHRMASSNAHMEN

- Serverseitige Daten
 - Enkodieren
 - Validieren
 - Nicht interpretieren
- Verschlüsselung verwenden
 - HTTPS
- Wichtige Transaktionen zusätzlich absichern
 - 2FA
- Whitelisting besser als Blacklisting

Speaker notes

- Blacklisting ist immer noch besser als nichts auszuschließen. Trotzdem besser auf Whitelisting setzen, wo es eben geht.

ALLGEMEINE ABWEHRMASSNAHMEN

- Minimalitätsprinzip
- Sicherheitsmaßnahmen nicht ausschalten/umgehen
 - Content Security Policy
 - Cross-Origin Resource Sharing
- Aktualisieren von Abhängigkeiten
 - Sonar - Dependency Check
 - Jenkins - Dependency Upgrade
- Weiterbildung/Fortbildung

Speaker notes

- Minimalitätsprinzip: Der Server sollte möglichst wenig Informationen preisgeben. Z.B. bei Fehlermeldungen.
- Sicherheitsmaßnahmen können bei der Entwicklung stören. Das heißt nicht, dass man sie ausschalten sollte.

PRAXIS

- Einbrechen in eine Beispielanwendung
 - `http://91.132.146.156:8000/login.php`
 - User: bee
 - Password: bug
- wer es lokal aufsetzen möchte
 - `dhbw_webengineering_2/security_bWAPP`
 - Wird mit "*docker-compose up -d*" gestartet
 - Docker wird benötigt

Speaker notes

- Durch die Anwendung "leiten" und die einzelnen Angriffsmöglichkeiten erklären.
- Sinnvoll ist sehr wahrscheinlich:
 - SQL Injection
 - Insecure DOR
 - Directory Traversal
 - Cross Site Scripting