Denne forelesningsøkten vil bli tatt opp og lagt ut i emnet i etterkant.

<u>Hvis</u> du ikke vil være med på opptaket:

	La være å delta med webkameraet ditt.
	La være å delta med mikrofonen din.
	Still spørsmål i Chat i stedet for som lyd. Hvis du ønsker kan spørsmålet også sendes privat til foreleser.

Høyskolen Kristiania

Høyskolen Kristiania

TK2100: Informasjonssikkerhet

11. Forelesning

Defensive Programming

Så langt

- CIA og AAA
- Kryptering
  - Symmetrisk (AES) vs Public Key (RSA)
  - Blokk (AES) vs strøm (RC4)
- Operativsystem
  - Sikring av prosess, minne, filsystem
  - Autentisering og autorisering + ACL
  - Bufferoverflows og «zero day attacks»
- Malware
  - Bakdører, logikkbomber; Virus, ormer og trojanere
  - Zero-day angrep; Rootkit; Botnet
  - Antivirus: Signatur, heuristikk

Så langt 2

- TCP/IP-modellen repetert
- Wireshark og sniffing
- Trusler på Link- & nettverks-laget
  - ARP-spoofing og –cache-poisioning
  - IP: IP-spoofing
  - ICMP: Ping flood, smurfe-angrep
- Trusler på Transportlaget
  - TCP: ACK-storm, SYN-flood, optimistisk ACK angrep («metningskotroll»-basert); sesjons-kidnapping
- Hovedsakelig ulike former for DoS
- Applikasjonslaget
  - DNS, DNS-spoofing, DNSSEC
- Brannmurer
  - Typer og teknikker (policies)
- Tunneling
  - SSH, IPSec og VPN
- Intrusion Detection («innbruddsalarmer»)
- Nmap og portscanning

Så langt 3

- Browseren («netteleseren») og sikkerhet på WWW
  - (dynamisk) HTML, HTTP, HTTP over TLS (https)
  - Sesjoner og cookies
- Angrep på klient
  - Sesjonskidnapping
  - Phishing, klick-hijacking
  - Media, addons og plugins
  - XSS og CSRF
- Angrep på tjener
  - Scripting og svakheter
  - SQL injisering
- Forsvar
- Lover og modeller
  - Personvern
  - Åndsverk
  - ISO 27000
  - Common Criteria
- Penetrasjonstesting
  - Hvorfor?
- Opphavsrett, patenter
- DRM
- Epost og spam

I dag

Høyskolen Kristiania

- Defensive Programming
- Penetrasjonstest funn

Høyskolen Kristiania

Defensive Programming

Principal of Defensive Programming

Høyskolen Kristiania

- General quality – reducing the number of software bugs and problems
- Making code comprehensive – the source code should be readable and understandable so it is approved in a code audit
- Making the software behave in a predictable manner despite unexpected input or user actions

- Vi vil fokusere i dette foredraget på det siste punktet; noen ganger referert til som et delsett som heter «sikker programmering»
- Mest fokus på dette i praksis i dag, ikke så mye terping på teorien og metodikken
- Vil også nevne begrepet "offensive programmering"

Defensive Programming

Høyskolen Kristiania

- Beskytt koden mot feil data «fra utsiden»
- Etabler sikre grensesnitt
- Valider alt ved input data gjennom grensesnittene; type, lengde, innhold i data, struktur på data
- Etabler en strategi for hvordan forskjellig typer feil data skal håndteres (erstatt med default data, fast fail, logging, osv)
- Ikke forvent at en ekstern metode kan kalles og vil oppføre seg slik den er dokumentert
- Kode for diagnosering, logging og tracing
- Forsiktig bruk av exception handling; exception handling skal ikke være «kode flyt» – men kritisk feilhåndtering

Avansert håndtering

Høyskolen Kristiania

- Kode som detekterer feil selv
- Kode som kan isolere skadelig data og beskytte seg selv og annen kode fra denne dataen
- Kode som kan gjenopprette seg selv

Eksterne metoder

Høyskolen Kristiania

- Ikke tro at eksterne metoder vil oppføre seg slik de er dokumentert
- Ikke tro at eksterne metoder vil oppføre seg i fremtiden slik de har oppført seg frem til nå

- Ikke vent «for alltid» på at en ekstern metode returnerer
  - «For alltid» er lenge hvis en tjeneste går ned…

- Forvent at returnerte verdier fra eksterne metoder kan være skadelige (metoden kan ha «blitt» hostile)

Pre conditions

Høyskolen Kristiania

- Pre conditions er en teknikk for å sikre at data er korrekte før de brukes
- Data sjekkes før de tas i bruk – dette gjør at det er lett å se i koden at alle input data er validert

```
public void CreateAppointment(DateTime dateTime)
{
    if (dateTime.Date < DateTime.Now.AddDays(1).Date)
        throw new ArgumentException("Date is too early");

    if (dateTime.Date > DateTime.Now.AddMonths(1).Date)
        throw new ArgumentException("Date is too late");

    /* Create an appointment */
}
```

Input validering

Høyskolen Kristiania

- All input må valideres
- Ikke stol på språket eller miljøet, selv om du kjører Azure Web-applikasjoner som har en viss beskyttelse innebygd - legg til spesifikke og spesialtilpasset inndatavalidering
- Ikke stol på en Web Application Firewall, dette er en ekstra forsvarslinje, så ikke forlat din første forsvarslinje (som er inndatavalidering)

- Hvis et inndatafelt inneholder et telefonnummer; BARE tillat tall og +-tegnet
- Hvis et inndatafelt inneholder et navn; BARE tillat tegn som vanligvis brukes i et navn på språkene du antar å støtte

Feil i vanlig input validering

Høyskolen Kristiania

- Inndatavalidering søker etter <script> og fjerner den spesifikke teksten:
  - Angrepsstrengen er: "test <scr<script>ipt> Alert ('pwned'); </script> "
  - Resultat etter sletting <script> blir en gyldig angrepsstreng
- Script tags er fjernet rekursivt (og riktig):
  - Angrepsstrengen er: "test <scr<script>ipt> Alert ('pwned'); </script> "
  - Resultat etter sletting blir “test Alert ('pwned'); </script>"
- Hva hvis angrepsstrengen er:
  - "test <IMG SRC=javascript:alert('pwned')> "

Whitelist ikke blacklist

Høyskolen Kristiania

- Det er nesten umulig, og veldig tidkrevende, å sjekke alle mulige måter å gi input som resulterer i at et skript blir kjørt
- Det samme gjelder andre angrep vi skal se på som SQL Injection og Code Injection

- Best Practice er derfor å begrense input ved å fjerne alle spesialtegn – eller helst fjerne alle tegn du ikke forventer!

- Input er et navn, tillat a-å, A-Å og ‘-’:
  - Angrepsstrengen er: "test <scr<script>ipt> Alert ('pwned'); </script> "
  - Resultat etter sletting blir “test scrscriptipt Alert pwned script“ – som aldri vil kunne skade noen…

Problemer med blacklist

Høyskolen Kristiania

- Problem med svartelisting er hva du skal svarteliste. Tenk deg at du ønsker å svarteliste en gitt IP-adresse 169.254.169.254
- På grunn av datamaskin heltall kan følgende IP-adresser (kanskje) være det samme:
  - 425.510.425.510
  - 2852039166
  - 7147006462
  - 0xA9.0xFE.0xA9.0xFE
  - 0xA9FEA9FE
  - 0x414141410A9FEA9FE
  - 0251.0376.0251.0376
  - 0251.00376.000251.0000376

XSS sårbarheter

Høyskolen Kristiania

- Den vanligste sårbarheten ved manglende input validering er XSS sårbarheter
- To typer;
  - Reflected XSS: HTTP Reply inneholder angrepsstrengen, og script utføres på siden når den vises i en browser
  - Stored XSS: Input lagres på en webside, og når websiden en gang senere lastes så blir script kjørt i en browser – dette kan være på en annen maskin enn den som utførte “angrepet”
- Reflected XSS betyr at en angriper må sende den som skal hackes en link som brukeren må trykke på
- Stored XSS betyr at en angriper kan angripe alle brukere av en webside (eksempel gjestebok fra forelesning WWW)

SQL Injection sårbarheter

Høyskolen Kristiania

- SQL sårbarheter forutsetter at input fra brukeren brukes til å bygge opp et SQL Statement

- SQL statements skal bygges opp av biblioteksfunksjoner, dette kalles «parameterized input»
- Aldri bygg opp en SQL statement ved å konkatenere strenger som inkluderer tekst fra brukeren
- “ SELECT * FROM Users WHERE Username = ‘ “ + txtUser.Text + “ ‘ AND Password = ‘ “ + txtPassword.Text + ““‘

- Hva skjer hvis txtUser blir satt til ‘ || 1 == 1; --
  - Resultat: SELECT * FROM Users WHERE Username = ‘’ || 1 == 1; --
- All bets are off :-)

Code/Command Injection sårbarheter

Høyskolen Kristiania

- Aldri bruk input fra en bruker for å bygge opp en kommando som skal kjøres på operativsystemet!

- Tenk et input felt «Kontroller om printer kjører» og en drop down liste med printernavn, en angriper ser på siden og oppdager at HTTP Request som sendes så oversettes navnene til en hardkodet liste over IP adresser
- Serveren kjører en lokal kommando «ping + list.IP_addr» i et CMD shell

- Angrepsstreng: «1.1.1.1 & format c:»
- Resultatet vil være at det kjøres to kommandoer:
  - Ping 1.1.1.1
  - Format c:

Hva er “input”

Høyskolen Kristiania

- En vanlig feil er å tenke på input som tekstfelter hvor brukeren kan taste inn data
- URL er også input
- HTTP Header felter er også input
- Cookie verdier er input – ikke stol på at det er den cookien du satt på siden din du får tilbake, det kan være et angrep

- Opplastinger av filer er det farligste
  - Innholdet i en fil kan være skadelig, enten for web serveren eller andre brukere
  - Fil NAVNET kan inneholde en script tag eller utføre SQL Injection :-)

Håndtere login – lagre passord

Høyskolen Kristiania

- Passord skal lagres backend som en hash
- IKKE lagre passord som klar tekst
- Ikke lagre passord kryptert
- Alltid bruk en salt verdi for å beskytte mot Rainbow tables
- Bruk PBKDF2 eller BCRYPT for å lagre passord; algoritmer som er laget for å ta lang tid – vanlige hash algoritmer er laget for å være kjappe…

- Det vanlige er å sende brukernavn og passord i klartekst, men over HTTPS (TLS 1.2)
- Ikke logg inn med hash (uten å være over TLS), det er sårbart ovenfor «send the hash» angrep
- For sikre applikasjoner anbefaler jeg å kryptere passordet

Logge en bruker inn på en tjeneste

Høyskolen Kristiania

- Hvis påloggingen skal være gyldig for mer enn ett miljø må du i utgangspunktet bruke et Single Sign On (SSO) system
- Om du logger deg på 2 eller 100 miljøer, bruker du den samme mekanismen
- SSO kan implementeres av JSON Web tokens eller en annen mekanisme mot en sentral database som AD eller LDAP

- I noen tilfeller er et alternativ å ikke ha brukerautentisering til den andre serveren, hvis det er mulig å sette opp en sikker mekanisme mellom de to serverne, vil brukeren da ikke ha tilgang til den andre serveren direkte – er vanligvis tilfellet mellom en frontend og en backend server, tilgang til backend vil da bli sikret med MTLS, OAuth2 + OpenID Connect eller lignende (ikke brukerens passord)

Logge inn på en backend database

Høyskolen Kristiania

- Passord for ressurser skal ikke være i kildekoden
- Utviklere bør ikke ha tilgang til produksjons passord
- Lagre et passord i miljøvariabler på serveren er OK, miljøvariabel config filer bør ikke være lagret til kildekontroll og ikke tilgjengelig for brukeren
- Lagring av passord i en sikker beholder, for eksempel Azure Key Vault, anbefales og anses som beste praksis

Rammeverk gir gratis hjelp, bruk det

Høyskolen Kristiania

- Hvis du har et Azure-miljø, kan du bruke Azure AD, både for brukerpålogging og for å sikre server-til-server-tilgang
- Skal du legge en skytjeneste i Azure, ikke bruk Azure Containers – da er du som utvikler ansvarlig for alt av sikkerhet, bruk Azure Web Services, Azure AD og Azure SQL
- Bruk AD for alt av autentisering og autorisasjon (ikke finn opp hjulet på nytt – bruker håndtering er vanskelig)

Brute force angrep

Høyskolen Kristiania

- En hver form for autentisering vil bli angrepet av en hacker, en hacker vil prøve flere typer passord angrep avhengig av hvor «lyst» han har til å komme inn i løsningen…
- En angriper starter først med de 1000 mest brukte passord, er passordet til EN av brukerne på systemet «password» eller «1234» så blir passordet funnet i løpet av noen minutter
- Så vil en angriper kjøre mer sofistikerte regel og hybrid angrep, en angriper kan ha scan mot brukere og passord kjørende i flere måneder før han «gir opp»
- En angriper som virkelig vil inn i en løsning vil til slutt prøve alle mulige kombinasjoner av bokstaver, tall og spesialtegn, det tar bare tid – men ofte for mange (tusen) år
- Problemet er at en angriper kan ha FLAKS

Brute force beskyttelse

Høyskolen Kristiania

- Hvis en brukers passord er forsøkt 3 ganger så vent i 10 sekunder med validering av passordet
- Hvis en brukers passord er forsøkt 4 ganger – 20 sekunder

  5 ganger – 40 sekunder

  6 ganger – 80 sekunder

  osv
- Hvis samme IP adresse har forsøkt å knekke passordet til forskjellige brukere 10 ganger – blokker all trafikk fra den IPen i 30 minutter

- Mer sofistikerte algoritmer har som formål å ikke være til hinder for en vanlig bruker som har glemt passordet, men stoppe brute force angrep (også distribuerte angrep) ved å øke tiden det tar å sjekke et passord

Secure session handling

Høyskolen Kristiania

- Session håndtering fører ofte til ulike sårbarheter, alle aspekter av sesjonshåndtering er for komplisert for dette eksempelet, men dette er en kort sjekkliste:

- Ikke mulig å ha flere sesjoner på samme bruker samtidig
- Samme sesjon og samme brukerkonto kan ikke nås fra ulike IP-adresser på samme tid
- Sesjon avsluttes automatisk etter en kort tidsavbruddsperiode (2 – 5 minutter)
- Bruk av session cookie ikke mulig etter at sesjon er avsluttet (etter at brukeren logger ut av programmet)

Offensiv Programmering

Høyskolen Kristiania

- Ekstreme tolkninger av Defensive Programming prinsippene tilsier at ingen funksjon / metode skal stole på noe data – alle data fra alle kilder skal antas å være skadelig kode
- Offensive Programming sier at man kun skal bruke Defensive prinsippene for data som mottas utenfra, men skal derimot stole «blindt» på alle data som kommer inne fra systemet / applikasjonen selv
- I Offensive Programming er det ansett som en fordel om en applikasjon krasjer ved interne feil i koden, for da er det lettere å finne feilen

- Dette betyr i praksis at hvis en database server kun skal kunne (by design) motta data fra en frontend server, da skal database serveren stole på alle data den får fra frontend serveren

Høyskolen Kristiania

Penetrasjonstest funn

(Ta lærdom av disse, feil gjort i virkeligheten!)

Penetrasjonstesting

Høyskolen Kristiania

- I en penetrasjonstest eller en Vulnerability Assessment forsøker man, på kortest mulig tid, å avdekke alle sårbarheter som KAN utnyttes i et angrep, eller som kan åpne for at et angrep er mulig
- Noen sårbarheter er så alvorlige at en angriper kan bruke dem direkte for å få full kontroll over systemet (kritiske sårbarheter)
- Ved å løse alle sårbarheter vil systemet bli sikrere, det betyr at en pen-tester vil påpeke alt fra design feil og «informational» funn – til kritiske sårbarheter

- Noen eksempler på faktiske funn i applikasjonstester:

XSS sårbarhet i input felter

Høyskolen Kristiania

- Kunden hadde et chat-system for å kommunisere med teknisk støtte
- Å skrive «Hei dette er <script> alert («Hacked»); </script> Bengt» inn i nevnte inntastingsfelt resulterte i en popup-boks fra nettleseren som sier «Hacked»

- Dette er den verste typen ikke-validert input; det er mange teknikker å omgå svake filtere, men å ikke ha noe filtere i det hele tatt viser en stor mangel på sikkerhetskunnskap

Begrense tilgang til IP – men FAIL

Høyskolen Kristiania

- Best Practice er å begrense sky tjenester beregnet for intern bruk til IP-adresser. Som oftest bedriftsnettverkets offentlige adresse
- De fleste selskaper har et gjestenettverk, i 99% av tilfellene har gjestenettverket samme offentlige IP-adresse (!)
- En angriper som har besøkt selskapet, eller er i stand til å bryte det super sikre passord "visit123" som er skrevet på den whiteboarden som er synlig fra vinduet, kan derfor omgå IP-filteret
- Aldri stole utelukkende på et IP-filter for å sikre tilgang til en tjeneste!

Usikre TLS ciphers

Høyskolen Kristiania

- Gjennom ulike pen-tester i 2020 og 2021 kan jeg fortsatt se TLS 1.0 støttet, og servere som støtter 3DES og til og med RC4-algoritmer, Diffie Hellman-nøkkelutveksling er også ansett som ikke sikker

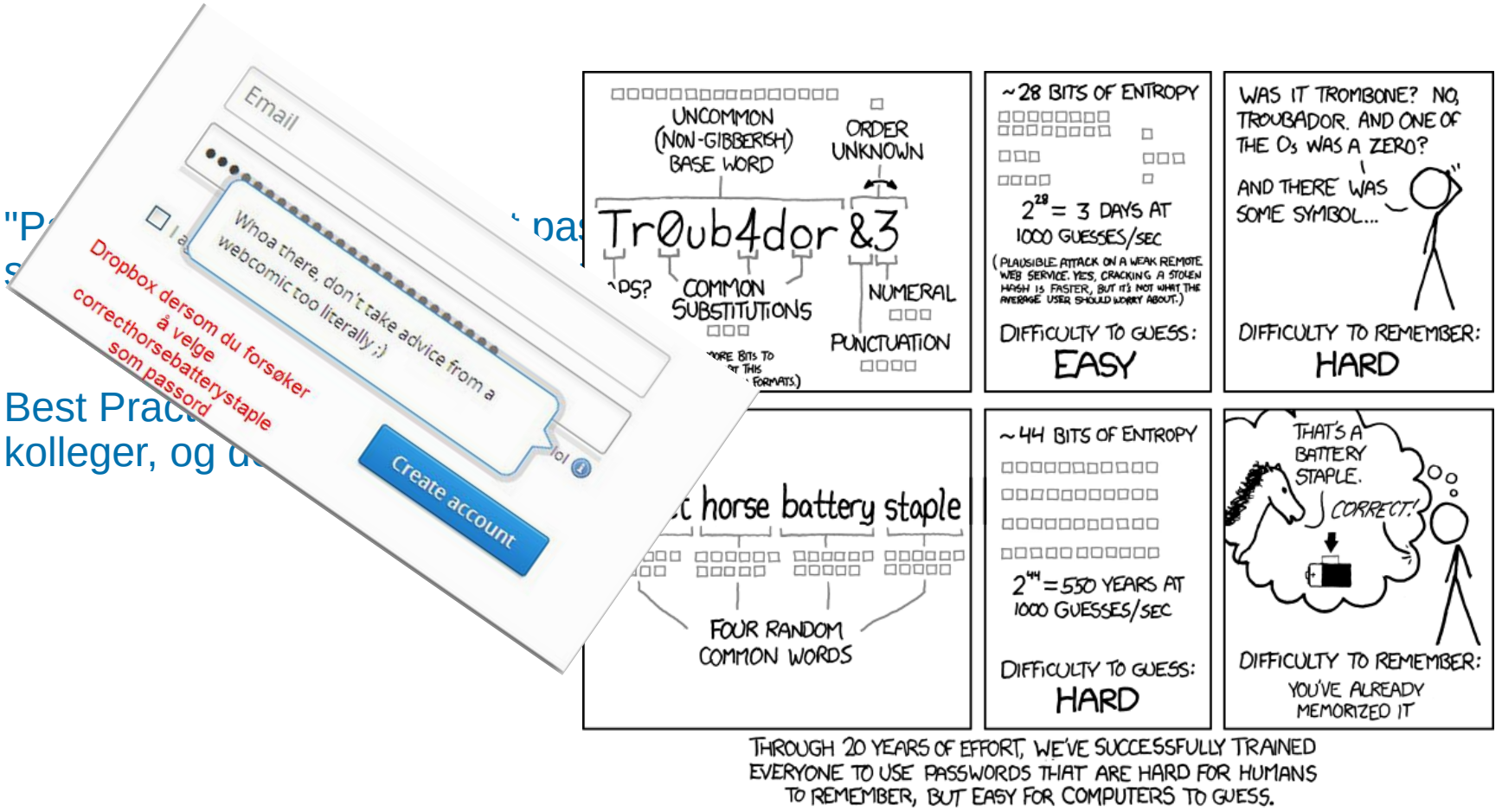
- Test alltid en webserver ved hjelp av:

  ssllabs.com

- Hvis du ikke får en A eller A+ rating, starter du på nytt og prøver igjen…

Svake passord

- "P… pas…
- Best Prac… kolleger, og d…



Stjele cookies og hijacke sesjoner

Høyskolen Kristiania

- Muligheten for å stjele Session Cookies er den vanligste sårbarheten i Web-applikasjoner
- En angriper får tak i informasjonskapselen (sårbarheten bryr seg ikke om hvordan), kopierer informasjonskapselen til en annen datamaskin og overtar en brukers sesjon

- Den verste sårbarheten jeg fant var der hvor en bruker hadde logget ut, men når en angriper gjenbrukt sesjons cookie valgte serveren å "gjenoppbygge" sesjonen basert på data i cookie...

Manglende Security Headers

Høyskolen Kristiania

- Moderne nettlesere har mye sikkerhet innebygd, du trenger bare å instruere leseren til å aktivere den
- Dette gjøres gjennom “Security Headers”
- Content Security Policy
- HTTP Strict Transport Security
- X-Content-Type-Options
- X-Xss-Protection
- X-Frame-Option

- https://www.owasp.org/index.php/OWASP_Secure_Headers_Project

Login over HTTP

Høyskolen Kristiania

- Vi er nå i 2021, bare bruk krypterte kanaler, all HTTP-trafikk må omadresseres til HTTPS (!)
- Hvis du logger på via en ukryptert HTTP er dette en alvorlig sårbarhet
- Husk at HTTP trafikk kan leses av i klartekst på switchen, ISPen din kan lese det (og logge det), hvis du er på et Wifi nett kan aksesspunktet ditt lese det – i teorien kan hvem som helst lese trafikken…
- Internett (TCP/IP) gir ingen form for sikkerhet, all sikkerhet på legges på applikasjonslaget, og det er det HTTPS skal

Beskyttet mot Brute Force – men ikke introduser nye feil!

Høyskolen Kristiania

- Vær svært forsiktig når man implementerer Brute Force beskyttelse, jeg har sett to forskjellige problemer introdusert av over-ivrige utviklere:

- Ikke lag et "denial of service" angrep ved å låse ut brukere permanent etter 3 mislykkede forsøk!
- Ikke lag et "bruker enumerering" angrep ved å gi forskjellige feilmeldinger hvis brukernavn eksisterer eller ikke!

Sårbar ovenfor DUHK angrep

Høyskolen Kristiania

- Passord og krypteringsnøkler hardkodet i kildekoden?
- Vanligvis er slike sårbarheter når Web-applikasjonen skal ha tilgang til en SQL-database backend, og autentiseringen for å få tilgang til backend er lagret i kildekoden
- Don’t Use Hardcoded Keys!

Session Handling ikke tilfredstillende

Høyskolen Kristiania

- Server som gjenoppretter brukers sesjon automatisk hvis en gammel session cookie blir gjenbrukt (etter en måned)

  FAIL

- Server som har en cookie som heter «logged-in : true» hvis brukeren (med cookie «user : Bengt» er logget inn

  HARD FAIL

- Server som har en cookie som viser seg å være BASE64 enkodet «BRUKERNAVN:RETTIGHETER:PASSORD»

  EPIC FAIL

OAuth 2.0 “bearer token” er ikke for autentisering

Høyskolen Kristiania

- OAuth 2.0 er et aksesstoken, ikke et autentiseringstoken
- Det er viktig for utviklere å vite hva OAuth kan og ikke kan brukes til, og hvilke andre mekanismer som kreves, ellers kan det føre til flere sårbarheter
- https://hueniverse.com/oauth-2-0-and-the-road-to-hell-8eec45921529

Upatchet servere

Høyskolen Kristiania

- Vi møter ofte servere som ikke er oppgradert

- Det er svært viktig å holde alle servere oppdatert, til enhver tid, gjennom hele levetiden til serveren
- Eksempel som ble funnet var en server i produksjonsmiljø (som ikke hadde blitt avdekket i de siste 3 pen-tester fordi andre testere bare hadde testet testmiljøet…):
  - Apache 2.4.6, from July 22th 2013; CVE-2017-7679, CVE-2018-1312, CVE-2017-15715, CVE-2014-0226, CVE-2017-9788

- Den største synderen er OpenSSH, det verste jeg har funnet er en installasjon fra 2004 – 17 år gammel…

Supportere HTTP metode TRACE

Høyskolen Kristiania

- Hvis en server støtter metoden TRACE gjør det serveren sårbar for et angrep kalt cross site tracing

- Les rapport om XST:
- http://www.cgisecurity.com/whitehat-mirror/WH-WhitePaper_XST_ebook.pdf

Bruke CSRF-token – men FAIL

Høyskolen Kristiania

- Mange kunder jeg tester er sårbare for Cross Site Request Forgery; et angrep der en kobling i en phishing e-post utnytter cookien nettleseren vil legge til all trafikk til en server
- En bruker som er pålogget, kan bli utsatt for en angriper som utnytter bruker sin påloggede økt til å utføre andre oppgaver

- Dette løses ved å legge til et URL-parameter eller et header felt med en hemmelig verdi

- En kunde hadde implementert et CSRF mottiltak i teorien, men på alle mislykkede HTTP Requests svarte serveren med en komplett nettside til kunden; inkludert skriptet som setter riktig CSRF token i headeren...

Begrense database tilgang – selv uten tilgang til internett

- En webserver kommuniserer ofte med en database server backend, det er sjeldent at databaseserveren er tilgjengelig fra Internett
- Uansett må serveren være beskyttet med autentiseringskrav

- Hos en kunde hadde de 3 miljøer; en TEST webserver, en QA webserver og produksjon Web server. I backend var det tre MongoDB database servere - en for test, en for QA og en for Prod
- Alle database tjenere der uten brukerautentisering – slik at en angriper som har brutt test web-tjeneren, kan fritt få tilgang til produksjonsdatabasen...

Beskytte servere, og introdusere sårbarheter

Høyskolen Kristiania

Tester av store konserner viser at sikkerhetssoftware ofte blir utdatert og introduserer nye sårbarheter

Eksempler;

- Tilgangskontroll software som beskyttet «sikker sone» inneholdt sårbarhet som tillot bypass av autentisering!
- Big-IP F5 web proxy inneholdt egne programmeringsfeil i skript språket som utgjorde reglene
- Iron Port server som skal beskytte epost server inneholdt mye mer alvorlige sårbarheter enn web serveren selv (som VAR patchet!)
- Cisco ASA server som var hacket av CIA og lekket ut av Shadow Brokers (når de hacket CIA…)

Adgangskontrollsystem kan styres over TELNET

Høyskolen Kristiania

- Hos et selskap fant jeg et Stanley Security adgangskontrollsystem
- Som eksponerte telnet på port 9999

- Jeg logget meg på, og så fort at jeg kunne fjern-administrere hele systemet deres uten brukernavn og passord…
- Turte faktisk ikke gjøre noe der, men det så ut som at jeg kunne legge til nye adgangskort!

Are you guys for real???

Høyskolen Kristiania

Hos et selskap jeg testet fant jeg noe merkelig

- Utdatert OpenSSH server versjon 4.3 – sett det før
- SendMail server versjon 8.13.8 – det var uvanlig
- Eksponert og sårbar fildeling på 139 og 445

- MiniServ versjon 0.01 på port 10000 – wait a minute…

Jeg fant en maskin hvor en ansatt hadde installert pwnOS, et test image for å LÆRE SEG PEN-TESTING

- Come on…

http://scx020c07c.blogspot.com/2012/09/privilege-escalation-pwnos.html

Og ikke glem den fysiske sikkerheten

Høyskolen Kristiania

- Hvis en dør har smekklås (som alle dører med nøkkelkort har), og den ikke har en deadbolt;
- Da er det ikke noen grunn til å holde den døra låst!
- Kort video av hvordan en JIM virker:

https://www.youtube.com/watch?v=_ImwWw0IQqw

- Virker banalt og virker sikkert ikke?
  - Jeg har åpnet dørene på 4 lokasjoner slik!
- Mer underholdende er Deviant Olam:

I'll let myself in

https://www.youtube.com/watch?v=rnmcRTnTNC8&t=2413s

Men, ikke alt er dårlig…

Høyskolen Kristiania

- Vi satt to stykker hos en norsk bank, oppdraget var å teste en mobil applikasjon
- Etter å ha kartlagt demo serverne som ble brukt backend av applikasjonen kjørte vi NMAP, Nikto og Nessus scan mot de

- Plutselig står det en mann bak meg og sier «Driver dere å skanner nettverket her?»

- Da er det greit å ha tillatelsen lett tilgjengelig! :-)

Epic win :-)

That’s all folks

Høyskolen Kristiania

Neste forelesning

Høyskolen Kristiania

- Repetisjon – tirsdag 26. april

- Og så er det eksamen
  - Forbered dere til eksamen som normalt
  - Løs de tidligere eksamensoppgavene som er lagt ut (2019, 2020, 2021)

  - Hjemmeeksamen krever mer åpne spørsmål hvor dere kan vise drøfting og at dere forstår helheten (slik 2020/2021 eksamen er), men 2020/2021 eksamen kanskje var litt «korte» (de hadde stort sett bare hatt digital undervisning, med noe forventet dårligere læringsutbytte) så forvent en mellomting mellom de to eksamenene

  - Bestått / ikke bestått = besvar ALLE oppgaver så godt dere kan, så går det sannsynligvis fint (men svar på 2 oppgaver og «ikke gidd mer», da kan du forberede deg på konteeksamen i august…)
  - **Krav for bestått: cirka 40 av 100 poeng**
  - Husk at eksamen er individuell, ingen samarbeid er tillatt

Høyskolen Kristiania

LYKKE TIL PÅ EKSAMEN

TAKK FOR ET GODT SEMESTER, DERE HAR VÆRT FLINKE OG ENGASJERTE STUDENTER!