Welcome to

2. Initial Overview of Software Security

KEA Kompetence OB2 Software Security 2019

Henrik Lund Kramshøj hlk@zencurity.com @kramse

Slides are available as PDF, kramse@Github

2-initial-overview-sw-security.tex in the repo security-courses

Plan for today

Subjects

- Design vs Implementation
- Common Secure Design Issues
- Poor Use of Cryptography
- Basic Cryptography introduction
- Symmetric Cryptosystems
- Data Encryption Standard (DES) / Advanced Encryption Standard (AES)
- Public Key Cryptography
- Stream and Block Ciphers
- Input Validation

Exercises

- sslscan scan various sites for TLS settings, Qualys SSLLabs
- Buffer Overflow 101

Reading Summary

Curriculum:

AoST chapter 2: How Vulnerabilities Get into All Software

Related resources:

Secure Programming for Linux and Unix HOWTO, David Wheeler
`https://dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf`
A Graduate Course in Applied Cryptography By Dan Boneh and Victor Shoup *https://toc.cryptobook.us/* `https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf`

Goals: Data Security

Nine principles of data security

(1) **Access control**—Each identifiable clinical record shall be marked with an access control list naming the people or groups of people who may read it and append data to it. The system shall prevent anyone not on the list from accessing the record in any way.

(2) **Record opening**—A clinician may open a record with herself and the patient on the access control list. When a patient has been referred she may open a record with herself, the patient, and the referring clinician(s) on the access control list.

(3) **Control**—One of the clinicians on the access control list must be marked as being responsible. Only she may change the access control list and she may add only other health care professionals to it.

(4) **Consent and notification**—The responsible clinician must notify the patient of the names on his record's access control list when it is opened, of all subsequent additions, and whenever responsibility is transferred. His consent must also be obtained, except in emergency or in the case of statutory exemptions.

(5) **Persistence**—No one shall have the ability to delete clinical information until the appropriate time has expired.

(6) **Attribution**—All accesses to clinical records shall be marked on the record with the name of the person accessing the record as well as the date and time. An audit trail must be kept of all deletions.

(7) **Information flow**—Information derived from record A may be appended to record B if and only if B's access control list is contained in A's.

(8) **Aggregation control**—Effective measures should exist to prevent the aggregation of personal health information. In particular, patients must receive special notification if any person whom it is proposed to add to their access control list already has access to personal health information on a large number of people.

(9) **Trusted computing base**—Computer systems that handle personal health information shall have a subsystem that enforces the above principles in an effective way. Its effectiveness shall be evaluated by independent experts.

Source: *Clinical system security: Interim guidelines*, Ross Anderson, 1996

Design vs Implementation

Software vulnerabilities can be divided into two major categories:

- Design vulnerabilities
- Implementation vulnerabilities

Even with a well-thought-out security design a program can contain implementation flaws.

Common Secure Design Issues

- Design must specify the security model's structure
  Not having this written down is a common problem
- Common problem AAA Authentication, Authorization, Accounting (book uses audited)
- Weak or Missing Session Management
- Weak or Missing Authentication
- Weak or Missing Authorization

Input Validation

Missing or flawed input validation is the number one cause of many of the most severe vulnerabilities:

- Buffer overflows - writing into control structures of programs, taking over instructions and program flow
- SQL injection - executing commands and queries in database systems
- Cross-site scripting - web based attack type
- Recommend centralizing validation routines
- Perform validation in secure context, controller on server
- Secure component boundaries

Weak Structural Security

Our book describes more design flaws:

- Large Attack surface
- Running a Process at Too High a Privilege Level, dont run everything as root or administrator
- No Defense in Depth, use more controls, make a strong chain
- Not Failing Securely
- Mixing Code and Data
- Misplaced trust in External Systems
- Insecure Defaults
- Missing Audit Logs

Secure Programming for Linux and Unix Howto

More information about systems design and implementation can be found in the free resource:

Secure Programming for Linux and Unix HOWTO, David Wheeler

`https://dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf`

Chapter 5. Validate All Input details input validation in the context of Unix programs

Chapter 6. Restrict Operations to Buffer Bounds (Avoid Buffer Overflow)

Chapter 7. Design Your Program for Security

Principle of Least Privilege

Definition 14-1 The *principle of least privilege* states that a subject should be given only those privileges that it needs in order to complete the task.

Also drop privileges when not needed anymore, relinquish rights immediately

Example, need to read a document - but not write.

Database systems can often provide very fine grained access to data

Principle of Least Authority

Definition 14-2 The *principle of least authority* states that a subject should be given only the authority that it needs in order to complete its task.

Closely related to principle of least privilege

Depend if there is distinction between *permission* and *authority*

Permission - what actions a process can take on objects directly

Authority - as determining what effects a process may have on an object, either directly or indirectly through its interactions with other processes or subsystems

Book uses the example of information flow, passing information to second subject that can write

Principle of Fail-Safe defaults

Definition 14-3 The *principle of fail-safe defaults* states that, unless a subject is given explicit access to an object, it should be denied access to that object.

Default access *none*

In firewalls default-deny - that which is not allowed is prohibited

Newer devices today can come with no administrative users, while older devices often came with default admin/admin users

Real world example, OpenSSH config files that come with `PermitRootLogin no`

Principle of Economy of Mechanism

Definition 14-4 The *principle of economy of mechanism* states that security mechanisms should be as simple as possible.

Simple $->$ fewer complications $->$ fewer security errors

Use WPA passphrase instead of MAC address based authentication

Principle of Complete Mediation

Definition 14-5 The *principle of complete mediation* requires that all accesses to objects be checked to ensure that they are allowed.

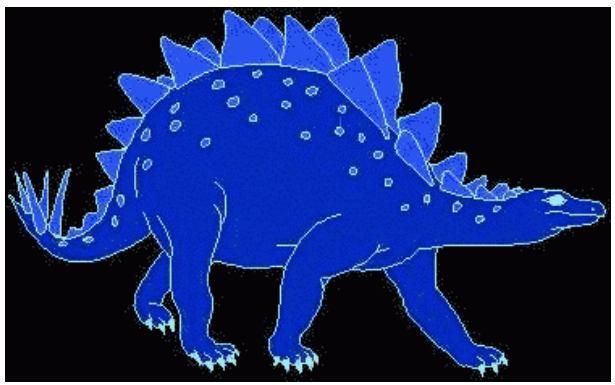
Always perform check

Time of check, time of use

Example Unix file descriptors - access check first, then can be reused in the future

Caching can be bad.

Principle of Open Design



Source: picture from `https://www.cs.cmu.edu/~dst/DeCSS/Gallery/Stego/index.html`

Definition 14-6 The *principle of open design* states that the security of a mechanism should not depend on the secrecy of its design or implementation.

Content Scrambling System (CSS) used on DVD movies

Mobile data encryption A5/1 key - see next page

Mobile data encryption A5/1 key

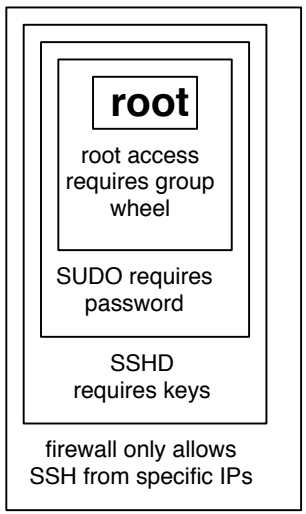
Real Time Cryptanalysis of A5/1 on a PC Alex Biryukov * Adi Shamir ** David Wagner ***

Abstract. A5/1 is the strong version of the encryption algorithm used by about 130 million GSM customers in Europe to protect the over-the-air privacy of their cellular voice and data communication. The best published attacks against it require between 240 and 245 steps. ... In this paper we describe new attacks on A5/1, which are based on subtle flaws in the tap structure of the registers, their noninvertible clocking mechanism, and their frequent resets. After a 248 parallelizable data preparation stage (which has to be carried out only once), the actual attacks can be **carried out in real time on a single PC.**

The first attack requires the output of the A5/1 algorithm during the first two minutes of the conversation, and computes the key in about one second. The second attack requires the output of the A5/1 algorithm during about two seconds of the conversation, and computes the key in several minutes. ... The approximate design of A5/1 was leaked in 1994, and the exact design of both A5/1 and A5/2 was reverse engineered by Briceno from an actual GSM telephone in 1999 (see [3]).

Source: `http://cryptome.org/a51-bsw.htm`

Principle of Separation of Privilege



Definition 14-7 The *principle of separation of privilege* states that a system should not grant permission based on a single condition.

Company checks, CEO fraud

Programs like *su* and *sudo* often requires specific group membership and password

Principle of Least Common Mechanism

Definition 14-8 The *principle of least common mechanism* states that mechanisms used to access resources should not be shared.

Minimize number of shared mechanisms and resources

Also mentions stack protection, randomization

Principle of Least Astonishment

Definition 14-9 The *principle of least astonishment* states that security mechanisms should be designed so that users understand the reason that the mechanism works they way it does and that using the mechanism is simple.

Security model must be easy to understand and targetted towards users and system administrators

Confusion may undermine the security mechanisms

Make it easy and as intuitive as possible to use

Make output clear, direct and useful
Exception user supplies wrong password, tell login failed but not if user or password was wrong

Make documentation correct, but the program best

Psychological acceptability - should not make resource more difficult to access

Qmail Security

> The qmail security guarantee In March 1997, I took the unusual step of publicly offering $500 to the first person to publish a verifiable security hole in the latest version of qmail: for example, a way for a user to exploit qmail to take over another account. My offer still stands. Nobody has found any security holes in qmail. I hereby increase the offer to $1000.

Some thoughts on security after ten years of qmail 1.0, Daniel J. Bernstein

- Started out of need and security problems in existing Sendmail
- Bug bounty early on. Donald Knuth has similar for his books

Qmail Security Paper, some answers

- Answer 1: eliminating bugs $->$ Enforcing explicit data flow, Simplifying integer semantics, Avoiding parsing
- Answer 2: eliminating code $->$ Identifying common functions, Reusing network tools, Reusing access controls, Reusing the filesystem
- Answer 3: eliminating trusted code $->$ Accurately measuring the TCB, Isolating single-source transformations, Delaying multiple-source merges, Do we really need a small TCB?

Qmail vs Postfix

> I failed to place any of the qmail code into untrusted prisons. Bugs anywhere in the code could have been security holes. The way that qmail survived this failure was by having very few bugs, as discussed in Sections 3 and 4.

Some thoughts on security after ten years of qmail 1.0, Daniel J. Bernstein

- This is NOT a comlete comparison of Qmail and Postfix `http://www.postfix.org/`!
- Postfix is comprised of many processes and modules. These modules typically are also chrooted and report back status only through very restricted interfaces
- It is also possible to turn off many components, allowing the system run with less code
- No Postfix program is setuid, all things are run by a master control process. A small setgid program used for mail submission - writing into the queue directory

Source: being a Postfix user and *Secure Coding: Principles and Practices* Eftir Mark Graff, Kenneth R. Van Wyk, June 2009

Vulnerability Analysis

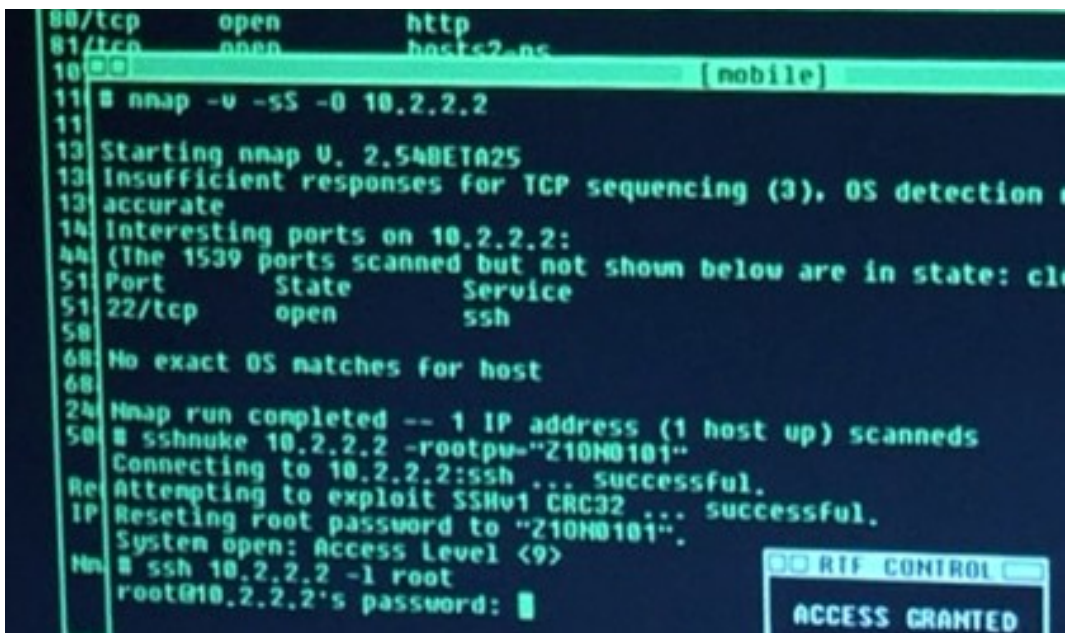
Vulnerability or security flaw

Exploiting the vulnerability happens by an attacker

A program or script used for this is called an *exploit*

Teknisk hvad er hacking

Trinity breaking in



Meget realistisk - sådan foregår det næsten:
`https://nmap.org/movies/`
`https://youtu.be/51lGCTgqE_w`

Hacking er magi

Hacking ligner indimellem magi

Hacking er ikke magi

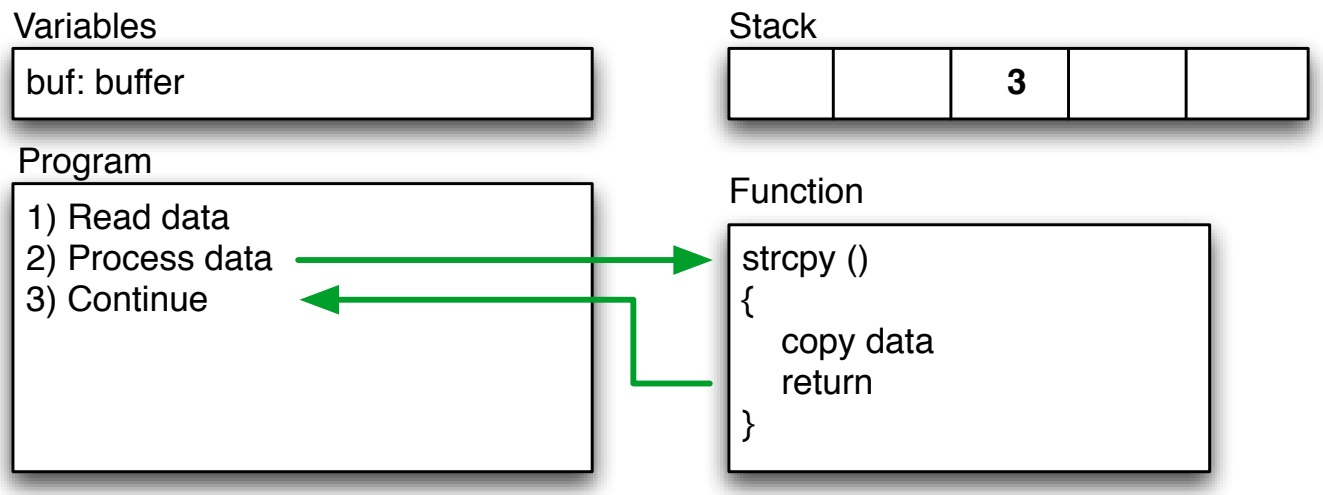
Hacking kræver blot lidt ninja-træning

Buffer overflows et C problem

Et buffer overflow er det der sker når man skriver flere data end der er afsat plads til i en buffer, et dataområde. Typisk vil programmet gå ned, men i visse tilfælde kan en angriber overskrive returadresser for funktionskald og overtage kontrollen.

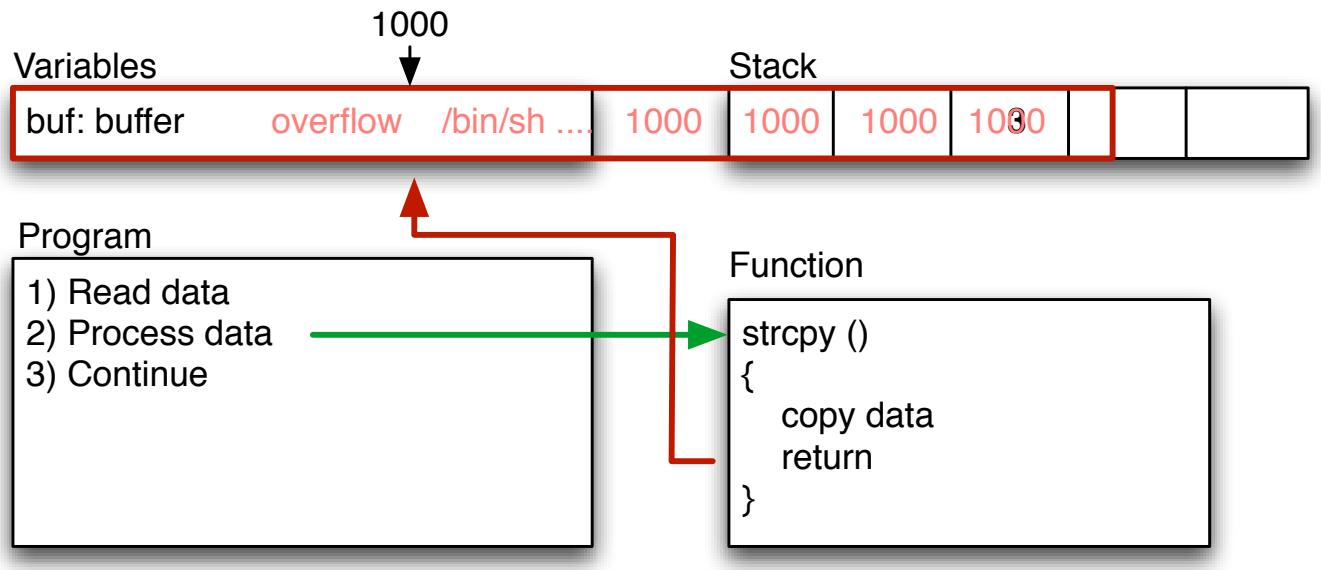
Stack protection er et udtryk for de systemer der ved hjælp af operativsystemer, programbiblioteker og lign. beskytter stakken med returadresser og andre variable mod overskrivning gennem buffer overflows. StackGuard og Propolice er nogle af de mest kendte.

Buffers and stacks, simplified




```
main(int argc, char **argv)
{       char buf[200];
         strcpy(buf, argv[1]);
         printf("%s\n",buf);
}
```

Overflow – segmentation fault



- Bad function overwrites return value!
- Control return address
- Run shellcode from buffer, or from other place

Exploits – udnyttelse af sårbarheder

- Exploit/exploitprogram er udnytter en sårbarhed rettet mod et specifikt system.
- Kan være 5 linier eller flere sider ofte Perl, Python eller et C program

Eksempel demo i Perl, uddrag:

```
$buffer = "";
$null = "\x00";
$nop = "\x90";
$nopsize = 1;
$len = 201; // what is needed to overflow, maybe 201, maybe more!
$the_shell_pointer = 0x01101d48; // address where shellcode is
# Fill buffer
for ($i = 1; $i < $len;$i += $nopsize) {
    $buffer .= $nop;
}
$address = pack('l', $the_shell_pointer);
$buffer .= $address;
exec "$program", "$buffer";
```

Hvordan finder man buffer overflow, og andre fejl

Black box testing

Closed source reverse engineering

White box testing

Open source betyder man kan læse og analysere koden

Source code review – automatisk eller manuelt

Fejl kan findes ved at prøve sig frem – fuzzing

Exploits virker typisk mod specifikke versioner af software

Privilegier privilege escalation

Privilege escalation er når man på en eller anden vis opnår højere privileger på et system, eksempelvis som følge af fejl i programmer der afvikles med højere privilegier. Derfor HTTPD servere på Unix afvikles som nobody – ingen specielle rettigheder.

En angriber der kan afvikle vilkårlige kommandoer kan ofte finde en sårbarhed som kan udnyttes lokalt – få rettigheder = lille skade

Eksempel: man finder exploit som giver kommandolinieadgang til et system som almindelig bruger

Ved at bruge en local exploit, Linuxkernen kan man måske forårsage fejl og opnå root, GNU Screen med SUID bit eksempelvis

Local vs. remote exploits

Local vs. remote angiver om et exploit er rettet mod en sårbarhed lokalt på maskinen, eksempelvis opnå højere privilegier, eller beregnet til at udnytter sårbarheder over netværk

Remote root exploit - den type man frygter mest, idet det er et exploit program der når det afvikles giver angriberen fuld kontrol, root user er administrator på Unix, over netværket.

Zero-day exploits dem som ikke offentliggøres – dem som hackere holder for sig selv. Dag 0 henviser til at ingen kender til dem før de offentliggøres og ofte er der umiddelbart ingen rettelser til de sårbarheder

Zero day 0-day vulnerabilities

Project Zero's team mission is to "make zero-day hard", i.e. to make it more costly to discover and exploit security vulnerabilities. We primarily achieve this by performing our own security research, but at times we also study external instances of zero-day exploits that were discovered "in the wild". These cases provide an interesting glimpse into real-world attacker behavior and capabilities, in a way that nicely augments the insights we gain from our own research.

Today, we're sharing our tracking spreadsheet for publicly known cases of detected zero-day exploits, in the hope that this can be a useful community resource:

Spreadsheet link: 0day "In the Wild"

`https://googleprojectzero.blogspot.com/p/0day.html`

- Not all vulnerabilities are found and reported to the vendors
- Some vulnerabilities are exploited *in the wild*

Demo: Insecure programming buffer overflows 101

- Small demo program `demo.c`
- Has built-in shell code, function `the_shell`
- Compile: `gcc -o demo demo.c`
- Run program `./demo test`
- Goal: Break and insert return address

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv)
{       char buf[10];
         strcpy(buf, argv[1]);
         printf("%s\n",buf);
}
int the_shell()
{  system("/bin/dash");  }
```

NOTE: this demo is using the dash shell, not bash - since bash drops privileges and won't work.

GDB GNU Debugger

GNU compileren og debuggeren fungerer ok, men check andre!

Prøv `gdb ./demo` og kør derefter programmet fra *gdb prompten* med `run 1234`

Når I således ved hvor lang strengen skal være kan I fortsætte med `nm` kommandoen – til at finde adressen på `the_shell`
Skriv `nm demo | grep shell`

Kunsten er således at generere en streng der er præcist så lang at man får lagt denne adresse ind på det *rigtige sted*.

Perl kan erstatte AAAAA således `` `perl -e "print 'A'x10"` ``

Debugging af C med GDB

Vi laver sammen en session med GDB

Afprøvning med diverse input

- `./demo langstrengsomgiverproblemerforprogrammethvorformon`
- gdb `demo` efterfulgt af run med parametre
  `run AAAAAAAAAAAAAAAAAAAAAAAAAAAA`

Hjælp:
Kompiler programmet og kald det fra kommandolinien med `./demo 123456...7689` indtil det dør ... derefter prøver I det samme i GDB

Hvad sker der? Avancerede brugere kan ændre `strcpy` til `strncpy`

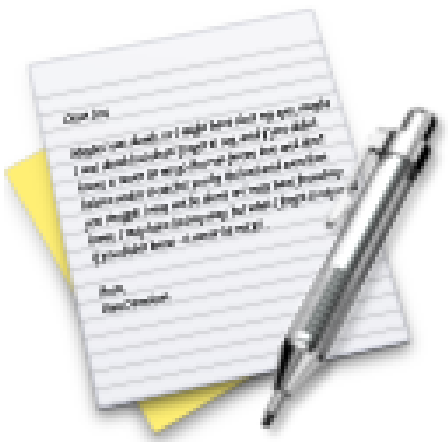
GDB output

```
hlk@bigfoot:demo$ gdb demo
GNU gdb 5.3-20030128 (Apple version gdb-330.1) (Fri Jul 16 21:42:28 GMT 2004)
Copyright 2003 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "powerpc-apple-darwin".
Reading symbols for shared libraries .. done
(gdb)  run AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Starting program: /Volumes/userdata/projects/security/exploit/demo/demo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Reading symbols for shared libraries . done
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Program received signal EXC_BAD_ACCESS, Could not access memory.
 0x41414140 in ?? ()
(gdb)
```

GDB output Debian 9 stretch

```
hlk@debian:~/demo$ gdb demo
GNU gdb (Debian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
...
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from demo...(no debugging symbols found)...done.
(gdb) run `perl -e "print 'A'x24"`
Starting program: /home/hlk/demo/demo `perl -e "print 'A'x24"`
AAAAAAAAAAAAAAAAAAAAAAAA

Program received signal SIGSEGV, Segmentation fault.
0x0000414141414141 in ?? ()
(gdb)
```

Exercise



Now lets do the exercise

Buffer Overflow 101 - 30-40min

which is number **9** in the exercise PDF.

WEP design major cryptographic errors

weak keying - 24 bit already known - 128-bit = 104 bit really

small initialisation vector (IV) - only 24 bit, every IV will be reused more often

CRC-32 integrity check NOT *strong* enough cryptographically

Authentication gives pad - if you get one *encryption pad* for one IV you can produce packets forever

Source: *Secure Coding: Principles and Practices*, Mark G. Graff and Kenneth R. van Wyk, O'Reilly, 2003

Poor Use of Cryptography

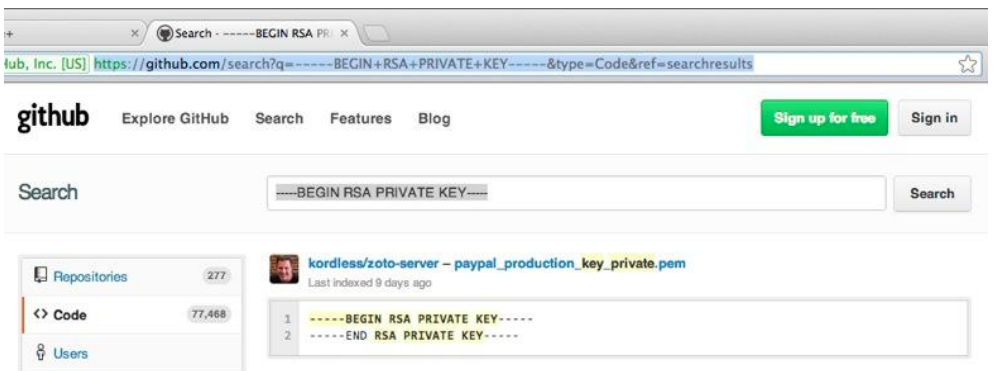
Common pitfalls

- Creating Your Own Cryptography
  Its easy to create something you cannot break, but that is not neccessarily secure
- Choosing the Wrong Cryptography - book recommend FIPS, lots of internet resources recommend NOT to use FIPS!
  Times changes, follow and read up
- Relying on Security by Obscurity
- Hard-Coded Secrets / Mishandling Private Information - if your mobile app binary contains a private key, and is being distributed to millions of users, is it really private - no

Cryptography is hard!

A Graduate Course in Applied Cryptography By Dan Boneh and Victor Shoup
`https://toc.cryptobook.us/`
`https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf`

Github Public passwords?



Sources:

https://twitter.com/brianaker/status/294228373377515522

http://www.webmonkey.com/2013/01/users-scramble-as-github-search-exposes-passwords-security-details/

http://www.leakedin.com/

http://www.offensive-security.com/community-projects/google-hacking-database/

Use different passwords for different sites, yes - every site!

Basic Cryptography introduction

Cryptography or cryptology is the practice and study of techniques for secure communication

Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary

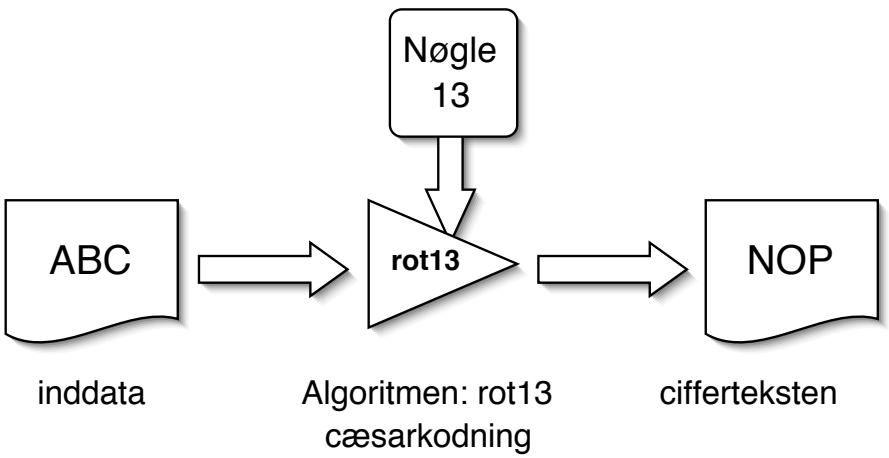
Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key, to ensure confidentiality, example algorithm AES

Public-key cryptography (like RSA) uses two related keys, a key pair of a public key and a private key. This allows for easier key exchanges, and can provide confidentiality, and methods for signatures and other services

Source: `https://en.wikipedia.org/wiki/Cryptography`

Encryption Decryption

Kryptografi

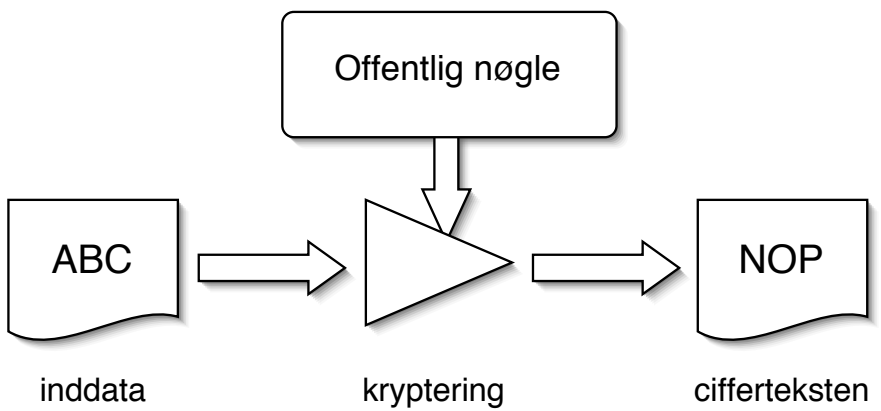


Kryptografi er læren om, hvordan man kan kryptere data

Kryptografi benytter algoritmer som sammen med nøgler giver en ciffertekst

- der kun kan læses ved hjælp af den tilhørende nøgle

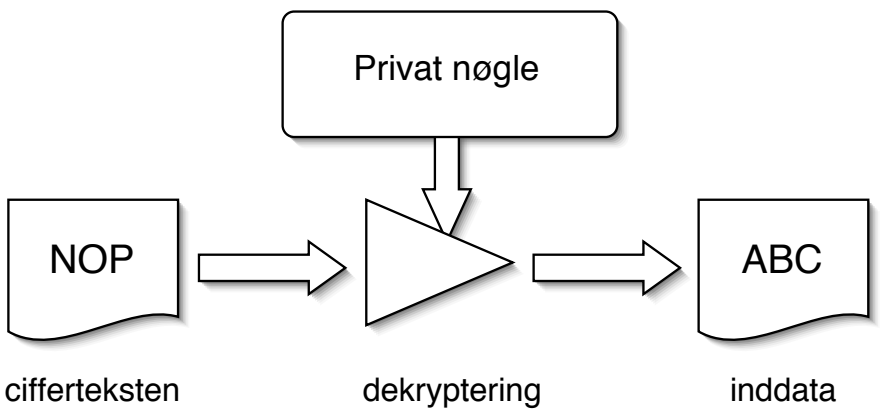
Public key kryptografi - 1



privat-nøgle kryptografi (eksempelvis AES) benyttes den samme nøgle til kryptering og dekryptering

offentlig-nøgle kryptografi (eksempelvis RSA) benytter to separate nøgler til kryptering og dekryptering

Public key kryptografi - 2



offentlig-nøgle kryptografi (eksempelvis RSA) bruger den private nøgle til at dekryptere

man kan ligeledes bruge offentlig-nøgle kryptografi til at signere dokumenter
- som så verificeres med den offentlige nøgle

NB: Kryptering alene sikrer ikke anonymitet

Kryptografiske principper

Algoritmerne er kendte

Nøglerne er hemmelige

Nøgler har en vis levetid - de skal skiftes ofte

Et successfuldt angreb på en krypto-algoritme er enhver genvej som kræver mindre arbejde end en gennemgang af alle nøglerne

Nye algoritmer, programmer, protokoller m.v. skal gennemgås nøje!

DES, Triple DES og AES

AES

Advanced Encryption Standard

DES kryptering - gammel og pensioneret!

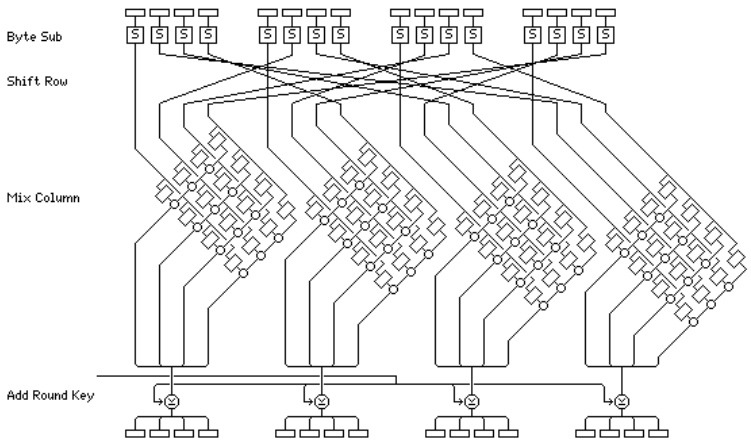
Der blev i 2001 vedtaget en ny standard algoritme Advanced Encryption Standard (AES) som afløser Data Encryption Standard (DES)

Algoritmen hedder Rijndael og er udviklet af Joan Daemen og Vincent Rijmen.

Se også `https://en.wikipedia.org/wiki/Advanced_Encryption_Standard`

Findes animationer (med fejl) `https://www.youtube.com/watch?v=mlzxpkdXP58`

AES Advanced Encryption Standard



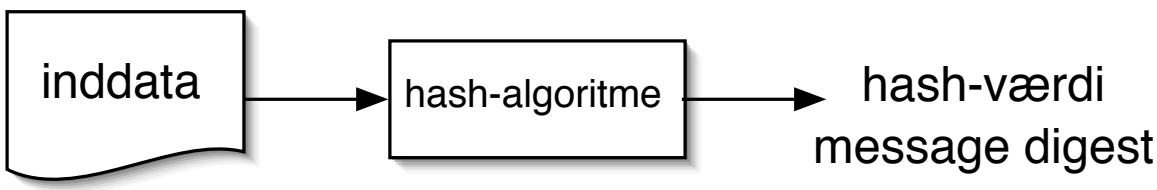
- The official Rijndael web site displays this image to promote understanding of the Rijndael round transformation [8].
- Key sizes 128,192,256 bit typical
- Some extensions in cryptosystems exist: XTS-AES-256 really is 2 instances of AES-128 and 384 is two instances of AES-192 and 512 is two instances of AES-256
- `https://en.wikipedia.org/wiki/RSA_(cryptosystem)`

RSA

> RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. ... In RSA, this asymmetry is based on the practical difficulty of the factorization of the product of two large prime numbers, the "factoring problem". The acronym RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1978.

- Key sizes 1,024 to 4,096 bit typical
- Quote from: `https://en.wikipedia.org/wiki/RSA_(cryptosystem)`

Hashing - MD5 message digest funktion



HASH algoritmer giver en næsten unik værdi baseret på input

værdien ændres radikalt selv ved små ændringer i input

MD5 er blandt andet beskrevet i RFC-1321: The MD5 Message-Digest Algorithm

Både MD5 og SHA-1 er idag gamle og skal ikke bruges mere

Idag benyttes eksempelvis `https://en.wikipedia.org/wiki/PBKDF2`

Encryption key length - who are attacking you

Encryption key lengths & hacking feasibility

Type of Attacker	*Budget*	Tool	*Time & Cost/Key 40 bit*	*Time & Cost/Key 56 bit*
Regular User	Minimal $400	Scavenged computer time FPGA	1 week 5 hours ($.08)	Not feasible 38 years ($5,000)
Small Business	$10,000	FPGA [1]	12 min.($.08)	556 days ($5,000)
Corporate Department	$300,000	FPGA ASIC [2]	24 sec. ($.08) 0.18 sec. ($.001)	19 days ($5,000) 3 hours ($38)
Large Corporation	$10M	ASIC	0.005 sec.($0.001)	6 min. ($38)
Intelligence Agency	$300M	ASIC	0.0002 sec.($0.001)	12 sec. ($38)

Source: `http://www.mycrypto.net/encryption/encryption_crack.html`

More up to date: In 1998, the EFF built Deep Crack for less than $250,000

`https://en.wikipedia.org/wiki/EFF_DES_cracker`

FPGA Based UNIX Crypt Hardware Password Cracker - 100 EUR in 2006

`http://www.sump.org/projects/password/`

Pass the hash

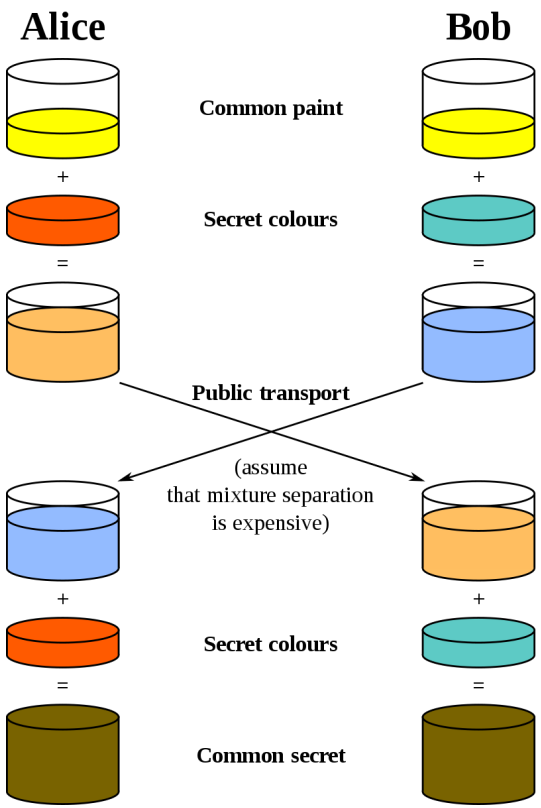
Lots of tools in pentesting pass the hash, reuse existing credentials and tokens *Still Passing the Hash 15 Years Later* `http://passing-the-hash.blogspot.dk/2013/04/pth-toolkit-for-kali-interim-status.html`

> If a domain is built using only modern Windows OSs and COTS products (which know how to operate within these new constraints), and configured correctly with no shortcuts taken, then these protections represent a big step forward.

Source:
`http://www.harmj0y.net/blog/penetesting/pass-the-hash-is-dead-long-live-pass-the-hash/` `https://samsclass.info/lulz/pth-8.1.htm`

Diffie Hellman exchange

Diffie–Hellman key exchange (DH)[nb 1] is a method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as originally conceptualized by Ralph Merkle and named after Whitfield Diffie and Martin Hellman.[1][2] DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography. ... The scheme was first published by Whitfield Diffie and Martin Hellman in 1976



- Quote from: `https://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange`
- Today we also use elliptic curves with DH
  `https://en.wikipedia.org/wiki/Elliptic-curve_cryptography`

Example Weak DH paper

Weak Diffie-Hellman and the Logjam Attack

Good News! Your browser is safe against the Logjam attack.

Diffie-Hellman key exchange is a popular cryptographic algorithm that allows Internet protocols to agree on a shared key and negotiate a secure connection. It is fundamental to many protocols including HTTPS, SSH, IPsec, SMTPS, and protocols that rely on TLS.

We have uncovered several weaknesses in how Diffie-Hellman key exchange has been deployed:

1. **Logjam attack against the TLS protocol.** The Logjam attack allows a man-in-the-middle attacker to downgrade vulnerable TLS connections to 512-bit export-grade cryptography. This allows the attacker to read and modify any data passed over the connection. The attack is reminiscent of the FREAK attack, but is due to a flaw in the TLS protocol rather than an implementation vulnerability, and attacks a Diffie-Hellman key exchange rather than an RSA key exchange. The attack affects any server that supports `DHE_EXPORT` ciphers, and affects all modern web browsers. 8.4% of the Top 1 Million domains were initially vulnerable.

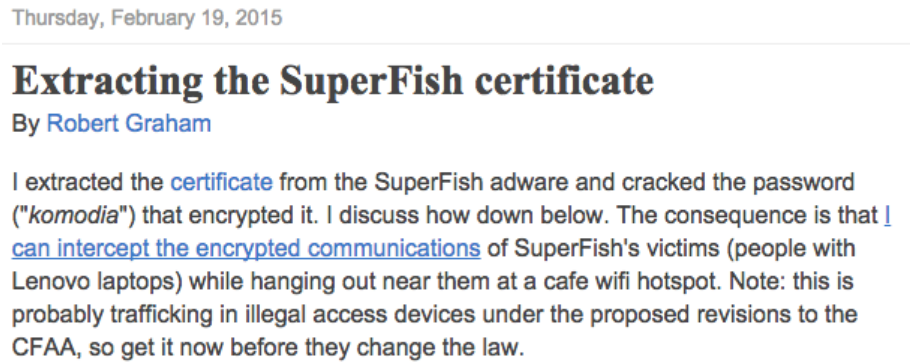
2. **Threats from state-level adversaries.** Millions of HTTPS, SSH, and VPN servers all use the same prime numbers for Diffie-Hellman key exchange. Practitioners believed this was safe as long as new key exchange messages were generated for every connection. However, the first step in the number field sieve—the most efficient algorithm for breaking a Diffie-Hellman connection—is dependent only on this prime. After this first step, an attacker can quickly break individual connections.

   We carried out this computation against the most common 512-bit prime used for TLS and demonstrate that the Logjam attack can be used to downgrade connections to 80% of TLS servers supporting `DHE_EXPORT`. We further estimate that an academic team can break a 768-bit prime and that a nation-state can break a 1024-bit prime. Breaking the single, most common 1024-bit prime used by web servers would allow passive eavesdropping on connections to 18% of the Top 1 Million HTTPS domains. A second prime would allow passive decryption of connections to 66% of VPN servers and 26% of SSH servers. A close reading of published NSA leaks shows that the agency's attacks on VPNs are consistent with having achieved such a break.

Source: `https://weakdh.org/` and `https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf`

Every year in different SSL/TLS implementations there have been problems.

Why?, because things like Superfish February 2015



Thursday, February 19, 2015

Extracting the SuperFish certificate

By Robert Graham

I extracted the certificate from the SuperFish adware and cracked the password ("*komodia*") that encrypted it. I discuss how down below. The consequence is that I can intercept the encrypted communications of SuperFish's victims (people with Lenovo laptops) while hanging out near them at a cafe wifi hotspot. Note: this is probably trafficking in illegal access devices under the proposed revisions to the CFAA, so get it now before they change the law.

Lenovo laptops included Adware, which did SSL/TLS Man in the Middle on connections. They had a root certificate installed on the Windows operating system, WTF!

Sources:

`https://en.wikipedia.org/wiki/Superfish`

`http://blog.erratasec.com/2015/02/extracting-superfish-certificate.html`

`http://www.version2.dk/blog/kibana4-superfish-og-emergingthreats-81610`

`https://www.eff.org/deeplinks/2015/02/further-evidence-lenovo-breaking-https-security-its-laptops`

Elliptic Curve

Public-key cryptography is based on the intractability of certain mathematical problems. Early public-key systems are secure assuming that it is difficult to factor a large integer composed of two or more large prime factors. For elliptic-curve-based protocols, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible: this is the **"elliptic curve discrete logarithm problem"(ECDLP)**. The security of elliptic curve cryptography depends on the ability to compute a point multiplication and the inability to compute the multiplicand given the original and product points. The size of the elliptic curve determines the difficulty of the problem.

Elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-EC cryptography (based on plain Galois fields) to provide equivalent security.[1]

- `https://en.wikipedia.org/wiki/Elliptic-curve_cryptography`
- Has very small key sizes

Transport Layer Security (TLS)

Oprindeligt udviklet af Netscape Communications Inc.

Secure Sockets Layer SSL er idag blevet adopteret af IETF og kaldes derfor også for Transport Layer Security TLS TLS er baseret på SSL Version 3.0

RFC-2246 The TLS Protocol Version 1.0 fra Januar 1999

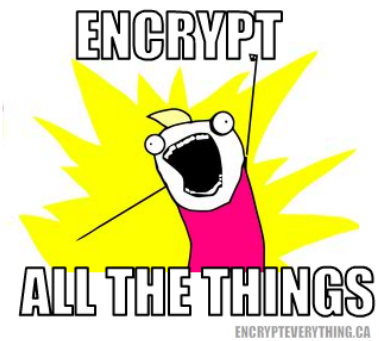
RFC-3207 SMTP STARTTLS

Det er svært!

Stanford Dan Boneh udgiver en masse omkring crypto
`https://crypto.stanford.edu/~dabo/cryptobook/`

TLS Server Name Indication extension

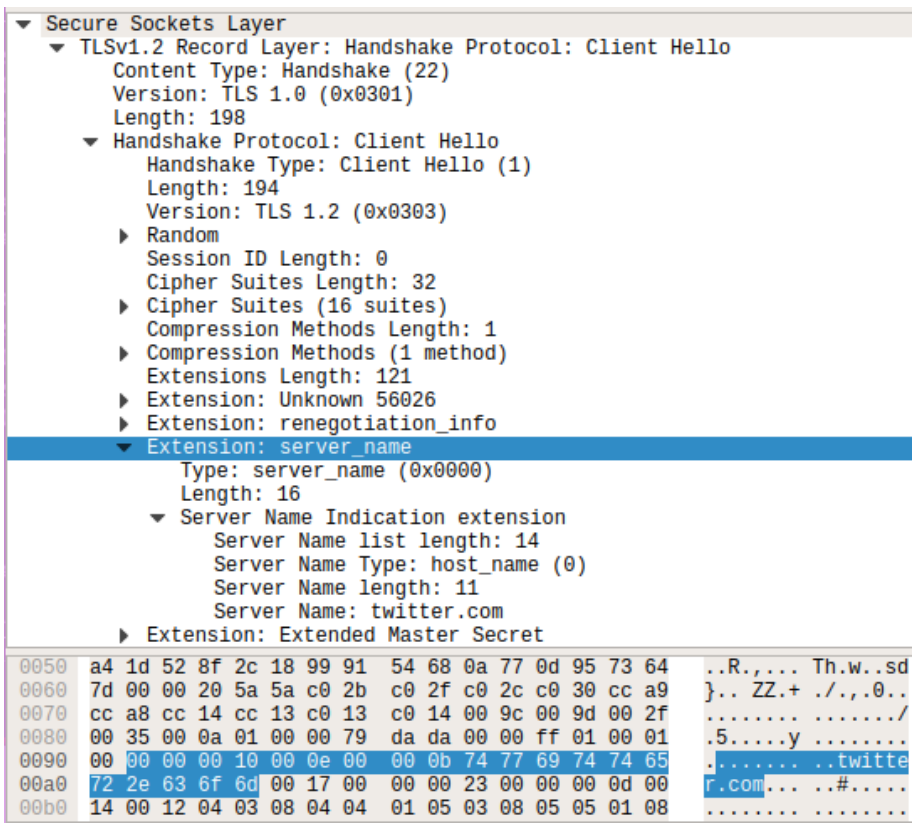
HTTPS skal der til!



Vi skal kryptere, men desværre så skjuler vores HTTPS ikke hvad site vi tilgår.

- HTTPS er idag TLS Transport Layer Security
- Verifikation sker med certifikater der præsenteres af server
- Der kan være flere sites på en enkelt IP - med SNI
- Desværre vælges det rigtige certifikat før krypteringen starter

TLS Server Name Indication example



Heartbleed CVE-2014-0160

The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

Nok det mest kendte SSL/TLS exploit

Source: `http://heartbleed.com/`

Heartbleed hacking

```
06b0: 2D 63 61 63 68 65 0D 0A 43 61 63 68 65 2D 43 6F  -cache..Cache-Co
06c0: 6E 74 72 6F 6C 3A 20 6E 6F 2D 63 61 63 68 65 0D  ntrol: no-cache.
06d0: 0A 0D 0A 61 63 74 69 6F 6E 3D 67 63 5F 69 6E 73  ...action=gc_ins
06e0: 65 72 74 5F 6F 72 64 65 72 26 62 69 6C 6C 6E 6F  ert_order&billno
06f0: 3D 50 5A 4B 31 31 30 31 26 70 61 79 6D 65 6E 74  =PZK1101&payment
0700: 5F 69 64 3D 31 26 63 61 72 64 5F 6E 75 6D 62 65  _id=1&card_numbe
0710: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX   r=4060xxxx413xxx
0720: 39 36 26 63 61 72 64 5F 65 78 70 5F 6D 6F 6E 74  96&card_exp_mont
0730: 68 3D 30 32 26 63 61 72 64 5F 65 78 70 5F 79 65  h=02&card_exp_ye
0740: 61 72 3D 31 37 26 63 61 72 64 5F 63 76 6E 3D 31  ar=17&card_cvn=1
0750: 30 39 F8 6C 1B E5 72 CA 61 4D 06 4E B3 54 BC DA  09.l..r.aM.N.T..
```

- Obtained using Heartbleed proof of concepts - Gave full credit card details
- "can XXX be exploited- yes, clearly! PoCs ARE needed
  without PoCs even Akamai wouldn't have repaired completely!
- The internet was ALMOST fooled into thinking getting private keys from Heartbleed was not possible - scary indeed.

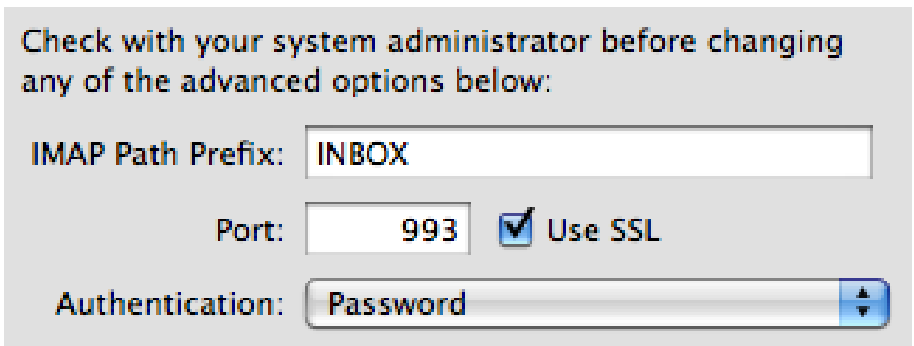
Key points after heartbleed

Source: picture source

`https://www.duosecurity.com/blog/heartbleed-defense-in-depth-part-2`

- Writing SSL software and other secure crypto software is hard
- Configuring SSL is hard
  check you own site `https://www.ssllabs.com/ssltest/`
- SSL is hard, finding bugs "all the time"`http://armoredbarista.blogspot.dk/2013/01/a-brief-chronology-of` `html`

SSL/TLS udgaver af protokoller



Mange protokoller findes i udgaver hvor der benyttes SSL

HTTPS vs HTTP

IMAPS, POP3S, osv.

Bemærk: nogle protokoller benytter to porte IMAP 143/tcp vs IMAPS 993/tcp

Andre benytter den samme port men en kommando som starter:

SMTP STARTTLS RFC-3207

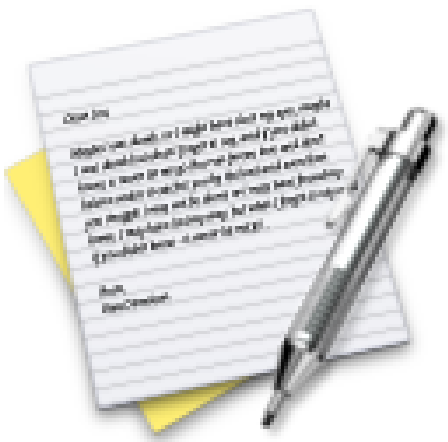
sslscan

```
root@kali:~# sslscan --ssl2 web.kramse.dk
Version: 1.10.5-static
OpenSSL 1.0.2e-dev xx XXX xxxx

Testing SSL server web.kramse.dk on port 443
...
  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength:    2048

Subject:  *.kramse.dk
Altnames: DNS:*.kramse.dk, DNS:kramse.dk
Issuer:   AlphaSSL CA - SHA256 - G2
```

Source: Originally sslscan from http://www.titania.co.uk but use the version on Kali

SSLscan can check your own sites, while Qualys SSLLabs only can test from hostname

Exercise

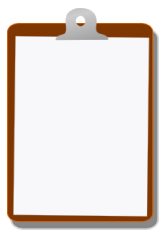


Now lets do the exercise

SSL/TLS scanners 15 min

which is number **10** in the exercise PDF.

For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books
Most days have less than 100 pages, but some days may have more!

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools