Velkommen til

Security Tools in Software Development

FOSS Aalborg

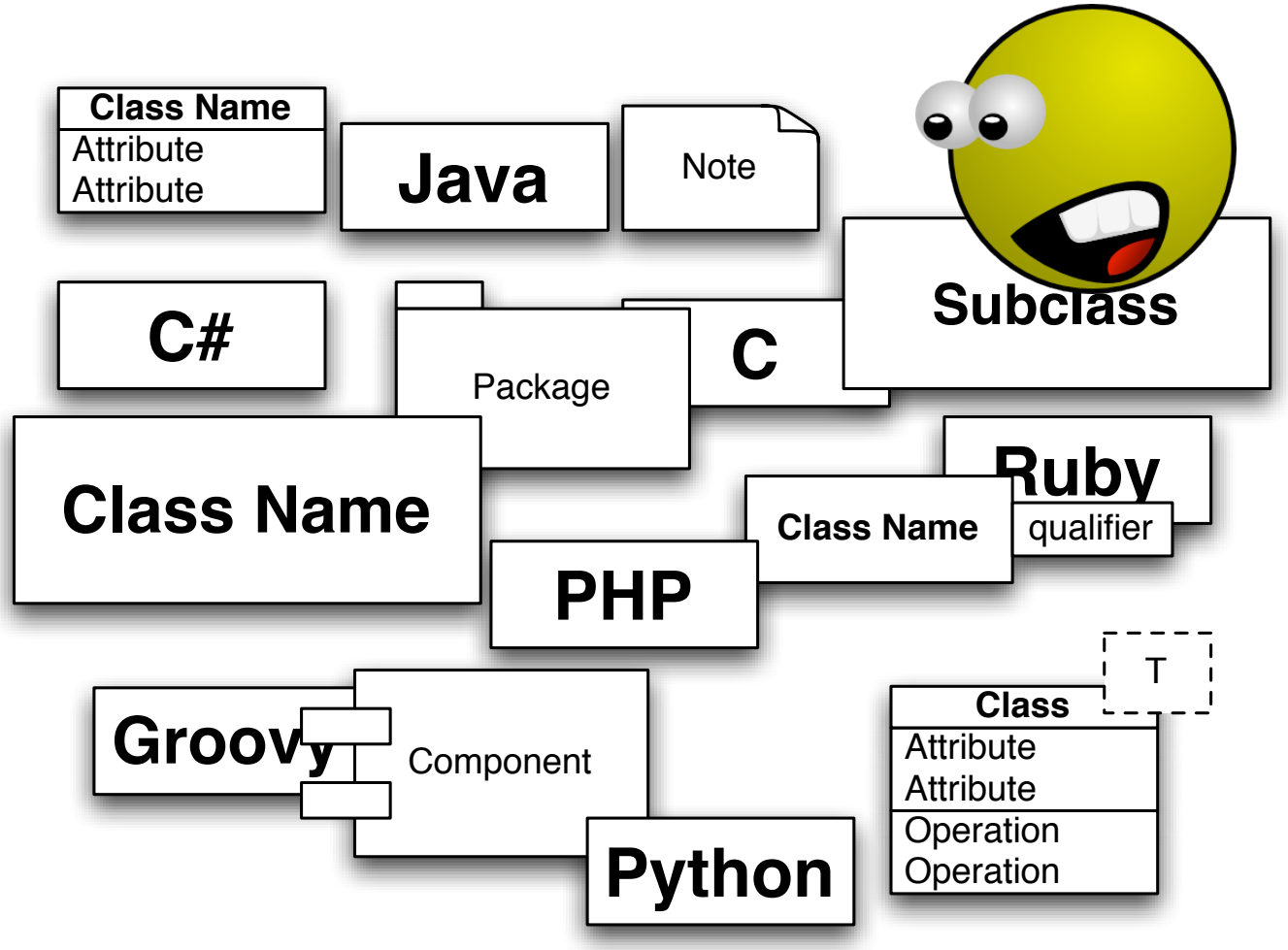
Henrik Lund Kramshøj
hlk@security6.net

http://www.security6.net

Slides are available as PDF and are in Danish only, sorry

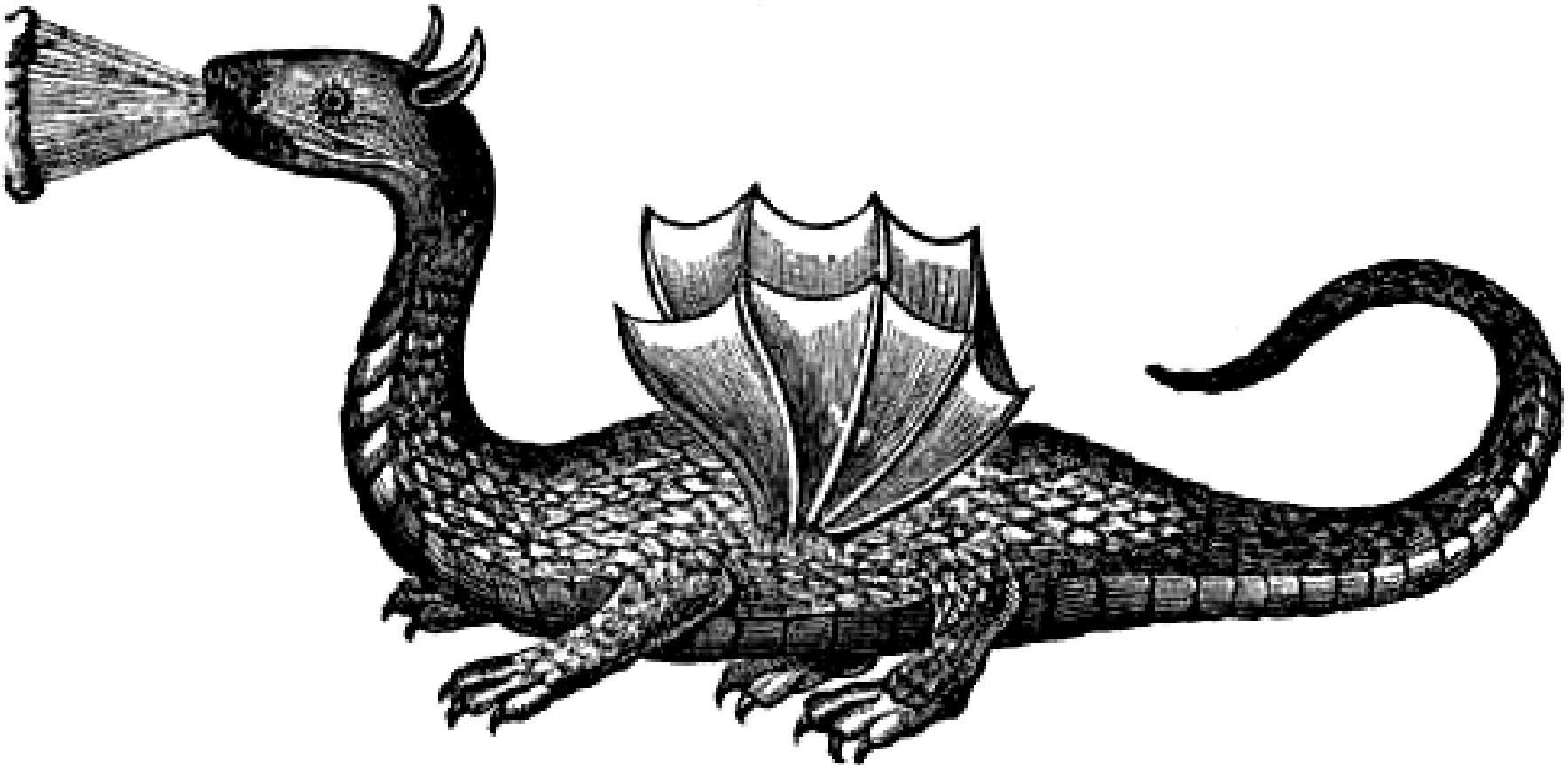
© copyright 2009 Security6.net, Henrik Lund Kramshøj

Formål



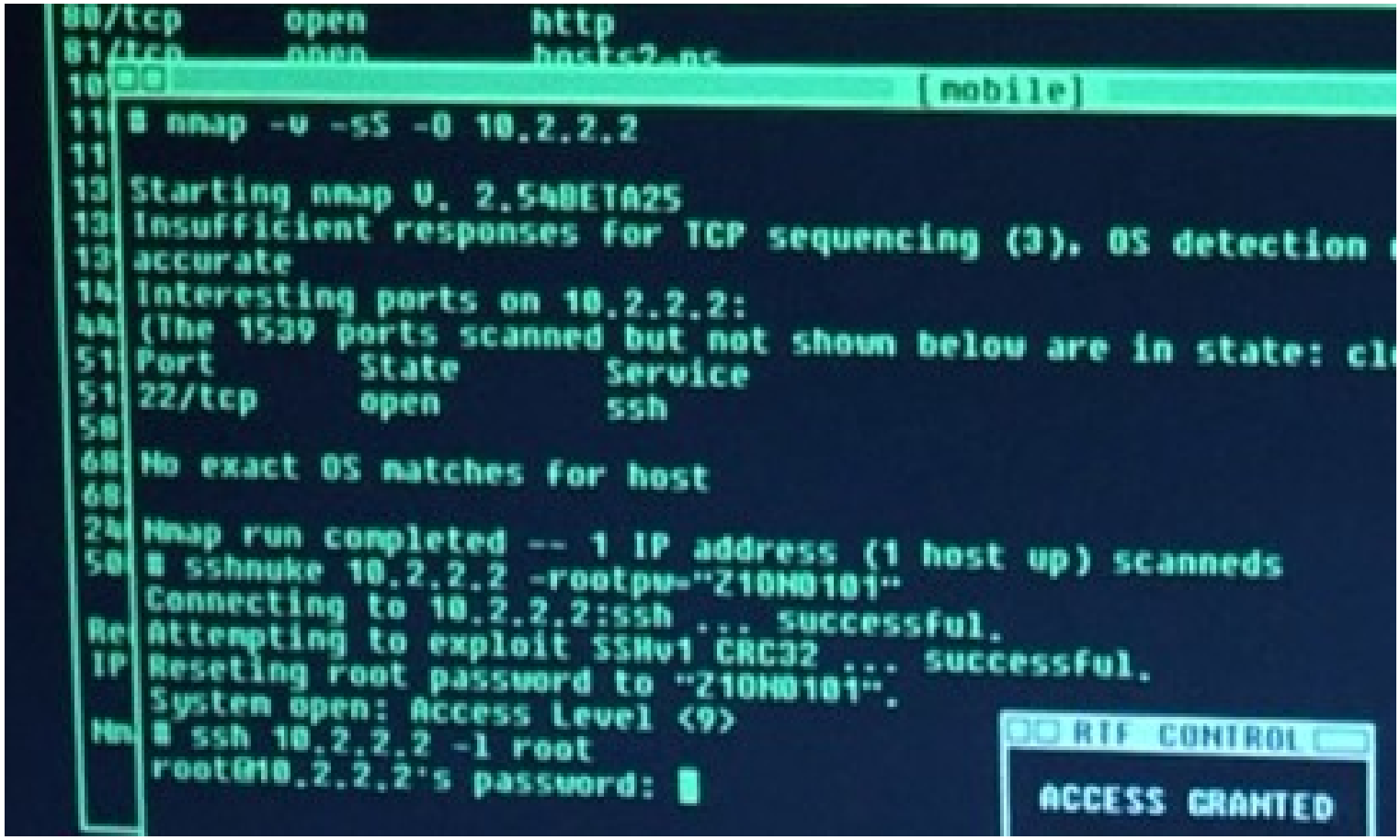
Lære om værktøjer der kan forbedre sikkerhed for produktionssystemer

Internet - Here be dragons



Matrix style hacking anno 2003

Trinity breaking in



http://nmap.org/movies.html

Meget realistisk http://www.youtube.com/watch?v=Zy5_gYu_isg

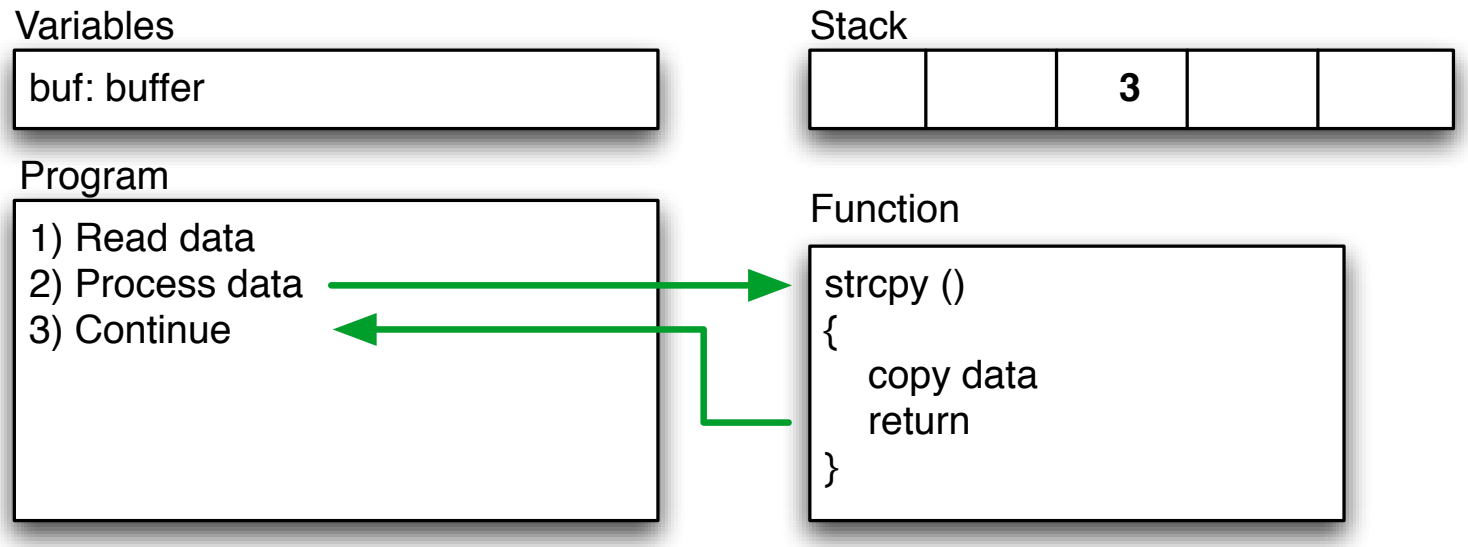
buffer overflows et C problem

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.

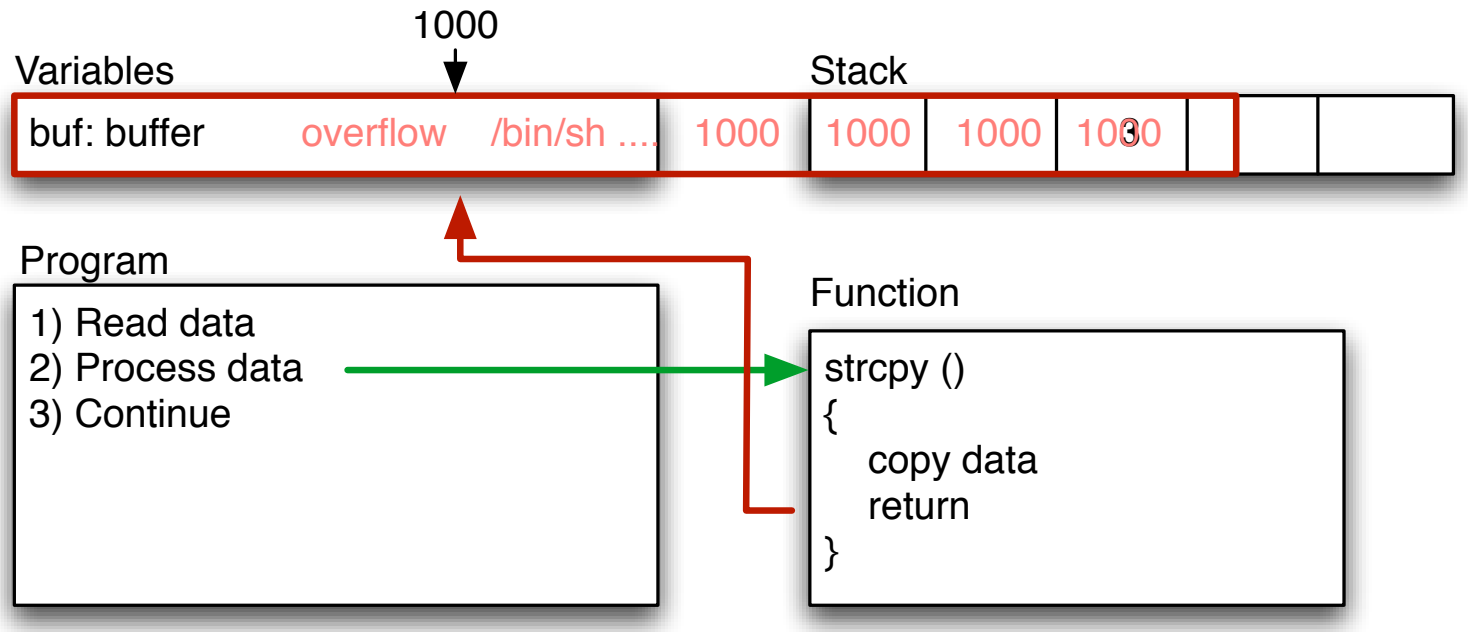
© copyright 2009 Security6.net, Henrik Lund Kramshøj

Buffer og stacks




```
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

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

Demo exploit in Perl

© copyright 2009 Security6.net, Henrik Lund Kramshøj

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

© copyright 2009 Security6.net, Henrik Lund Kramshøj

Forudsætninger

Bemærk: alle angreb har forudsætninger for at virke

Et angreb mod Telnet virker kun hvis du bruger Telnet

Et angreb mod Apache HTTPD virker ikke mod Microsoft IIS

Kan du bryde kæden af forudsætninger har du vundet!

Eksempler på forudsætninger

Computeren skal være tændt

Funktionen der misbruges skal være slået til

Executable stack

Executable heap

Fejl i programmet

alle programmer har fejl

Software udvikling er nemt

- **Du skal blot skrive perfekt kode første gang :-)**
- **Sikkerhed er svært**

Det er svært at skrive perfekt kode, om ikke umuligt

Så nu vil vi snakke om værktøjer til at forbedre situationen

© copyright 2009 Security6.net, Henrik Lund Kramshøj

Part 1 Low hanging fruits - easy

Højere kvalitet er mere sikkert

© copyright 2009 Security6.net, Henrik Lund Kramshøj

Coding standards - style

This file specifies the preferred style for kernel source files in the OpenBSD source tree. It is also a guide for preferred user land code style. These guidelines should be followed for all new code. In general, code can be considered “new code” when it makes up about 50more of the file(s) involved. ...
Use queue(3) macros rather than rolling your own lists, whenever possible. Thus, the previous example would be better written:

```
#include <sys/queue.h>
struct  foo {
LIST_ENTRY(foo) link;  /* Queue macro glue for foo lists */
            struct  mumble amumble; /* Comment for mumble */
            int     bar;
};
LIST_HEAD(, foo) foohead;     /* Head of global foo list */
```

OpenBSD style(9)

© copyright 2009 Security6.net, Henrik Lund Kramshøj

Coding standards functions

The following copies as many characters from input to buf as will fit and NUL terminates the result. Because strncpy() does not guarantee to NUL terminate the string itself, it must be done by hand.

```
char buf[BUFSIZ];

(void)strncpy(buf, input, sizeof(buf) - 1);
buf[sizeof(buf) - 1] = '\0';
```

Note that strlcpy(3) is a better choice for this kind of operation. The equivalent using strlcpy(3) is simply:

```
(void)strlcpy(buf, input, sizeof(buf));
```

OpenBSD strcpy(9)

© copyright 2009 Security6.net, Henrik Lund Kramshøj

Compiler warnings - gcc -Wall

```
$ gcc -o demo demo.c
demo.c: In function main:
demo.c:4: warning: incompatible implicit declaration of built-in
function strcpy

$ gcc -Wall -o demo demo.c
demo.c:2: warning: return type defaults to int
demo.c: In function main:
demo.c:4: warning: implicit declaration of function strcpy
demo.c:4: warning: incompatible implicit declaration of built-in
function strcpy
demo.c:5: warning: control reaches end of non-void function
```

Easy to do!

© copyright 2009 Security6.net, Henrik Lund Kramshøj

No warnings = no errors?

```
$ cat demo2.c
#include <strings.h>
int main(int argc, char **argv)
{
    char buf[200];
    strcpy(buf, argv[1]);
    return 0;
}
$ gcc -Wall -o demo2 demo2.c
```

Der er stadig alvorlige fejl!

Version control

Versionsstyring og configuration management har mange fordele

Hvem ændrede, hvornår og hvad

Hvorfor blev der foretaget en ændring

Med versionsstyring kan pre-commit hooks implementeres

Subversion sample hooks scripts

pre-commit - check

- case-insensitive.py
- check-mime-type.pl
- commit-access-control.pl
- commit-block-joke.py
- detect-merge-conflicts.sh
- enforcer
- log-police.py
- pre-commit-check.py
- svnperms.py
- verify-po.py

http://subversion.tigris.org/tools_contrib.html

http://svn.collab.net/repos/svn/trunk/contrib/hook-scripts/

Eksempel Enforcer

In a Java project I work on, we use log4j extensively. Use of System.out.println() bypasses the control that we get from log4j, so we would like to discourage the addition of println calls in our code.

We want to deny any commits that add a println into the code. The world being full of exceptions, we do need a way to allow some uses of println, so we will allow it if the line of code that calls println ends in a comment that says it is ok:

```
System.out.println("No log4j here"); // (authorized)
```

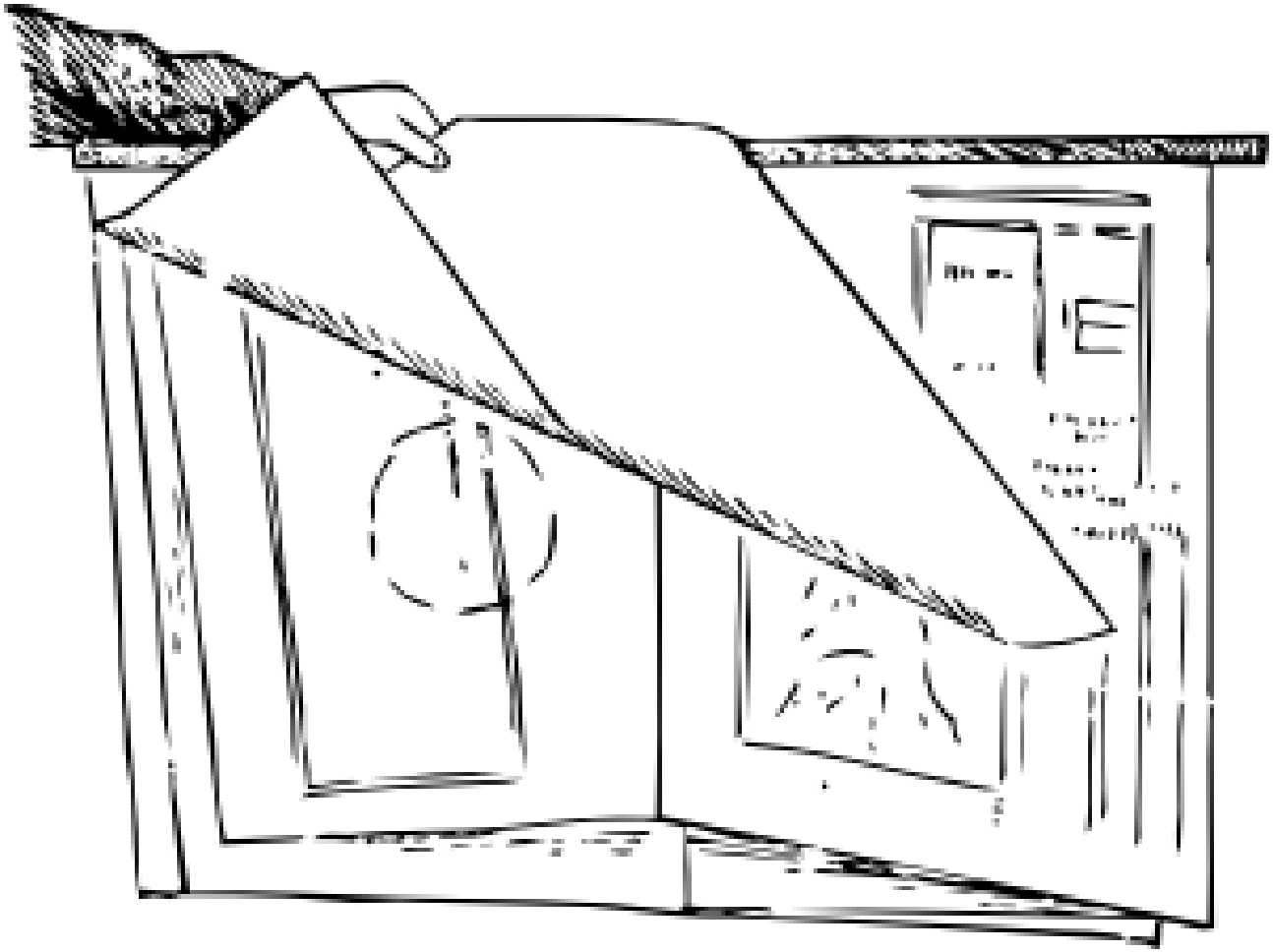
http://svn.collab.net/repos/svn/trunk/contrib/hook-scripts/enforcer/enforcer

Eksempel verify-po.py

```
#!/usr/bin/env python
"""This is a pre-commit hook that checks whether the contents
of PO files committed to the repository are encoded in UTF-8.
"""
```

http://svn.collab.net/repos/svn/trunk/tools/hook-scripts/verify-po.py

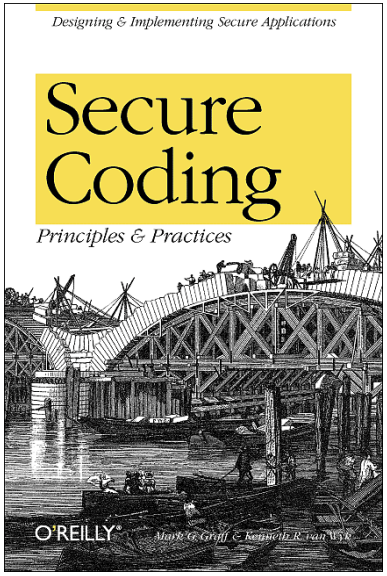
Part 2 Design for security - more work



Sikkerhed er kun effektivt hvis det tænkes ind i design

Secure Coding begynder med design

Secure Coding: Principles and Practices af Mark G. Graff, Kenneth R. Van Wyk 2003



Architecture/design while you are thinking about the application

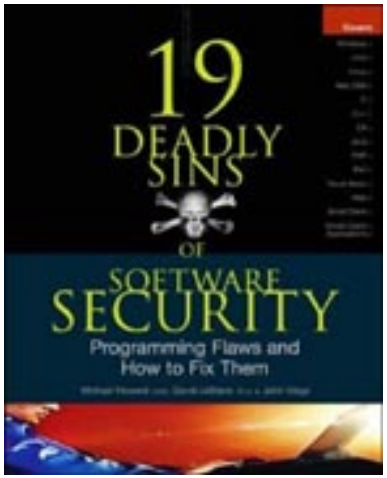
Implementation while you are writing the application

Operations After the application is in production

Ca. 200 sider, men tætpakket med information.

Sins in Software Security

19 Deadly Sins of Software Security af Michael Howard, David Leblanc, John Viega 2005



Obligatorisk læsning for alle udviklere

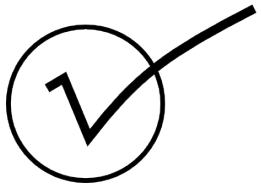
Forfatterne har skrevet mange gode bøger både før og efter

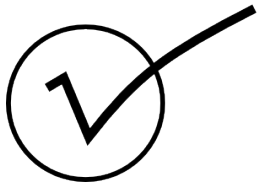
Denne bog er præcis og giver overblik

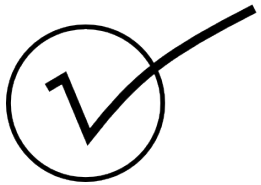
Ca. 270 sider, let at læse.

Buffer Overruns, Format String Problems, Integer Overflows, SQL Injection, Command Injection, Failing to Handle Errors, Cross-Site Scripting, Failing to Protect Network Traffic, Magic URLs Hidden Form Fields, Improper Use of SSL and TLS, Weak Password-Based Systems, Failing to Store and Protect Data Securely, Information Leakage, Improper File Access, Trusting Network Name Resolution, Race Conditions, Unauthenticated Key Exchange, Cryptographically Strong Random Numbers, Poor Usability

Part 3 Testing - more work now, less work in the long run


Test1


Test2


Test3

Test4

Højere kvalitet er mere sikkert

Hvorfor teste

Finde fejl under udviklingen af software

Sikre at software overholder krav til kvalitet

Finde fejl senere!

Undgå at gamle fejl optræder igen!

Test ofte

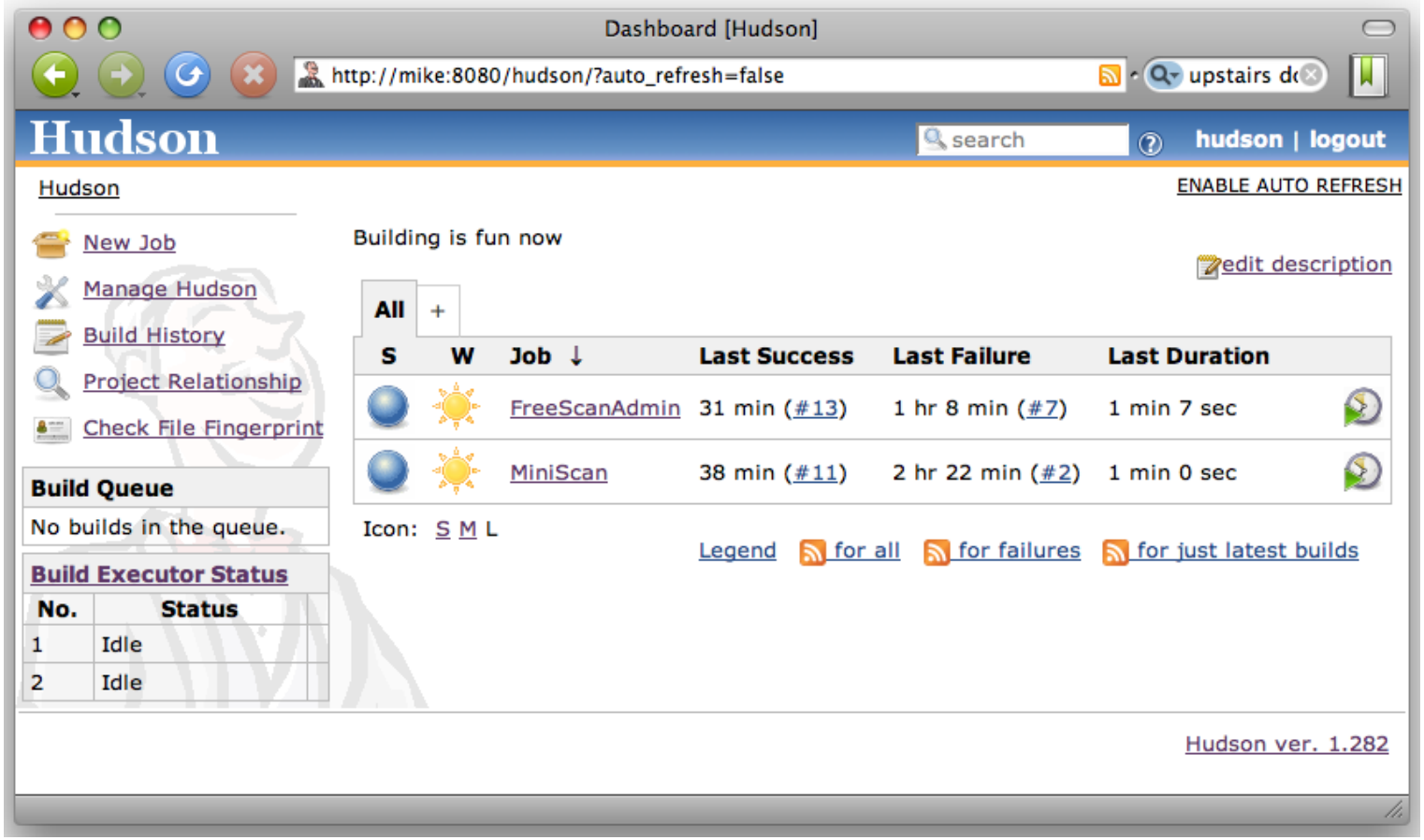
Unit testing - laveste niveau

```
public class TestAdder {
    public void testSum() {
        Adder adder = new AdderImpl();
        assert(adder.add(1, 1) == 2);
        assert(adder.add(1, 2) == 3);
        assert(adder.add(2, 2) == 4);
        assert(adder.add(0, 0) == 0);
        assert(adder.add(-1, -2) == -3);
        assert(adder.add(-1, 1) == 0);
        assert(adder.add(1234, 988) == 2222);
    }
}
```

Kan bruges til at teste enkelte dele af en applikation

Eksempel fra http://en.wikipedia.org/wiki/Unit_testing

Hudson and friends



Continous building and testing

Finder løbende fejl - hurtig feedback

Part 4 Analysis

Brug al den hjælp du kan til at finde fejl

Typer af analyse

statisk analyse

finder fejl uden at køre programmet

typisk findes konstruktioner som indeholder fejl, brug af forkerte funktioner m.v.

dynamisk analyse

findes ved at køre programmet, typisk i et specielt miljø

Statiske analyseværktøjer

Flawfinder http://www.dwheeler.com/flawfinder/

RATS Rough Auditing Tool for Security, C, C++, Perl, PHP and Python

PMD static ruleset based Java

http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

A Fool with a Tool is still a Fool

1. Run tool
2. Fix problems
3. Rinse repeat

Fixing problems?

```
char tmp[256]; /* Flawfinder: ignore */
strcpy(tmp, pScreenSize); /* Flawfinder: ignore */
```

Eksempel fra http://www.dwheeler.com/flawfinder/

PMD static ruleset based Java source code analyzer

PMD

PMD scans Java source code and looks for potential problems like:

- Possible bugs - empty try/catch/finally/switch statements
- Dead code - unused local variables, parameters and private methods
- Suboptimal code - wasteful String/StringBuffer usage
- Overcomplicated expressions - unnecessary if statements, for loops that could be while loops
- Duplicate code - copied/pasted code means copied/pasted bugs

You can **download everything from here**, and you can get an overview of all the rules at the rulesets index page.

PMD is integrated with JDeveloper, Eclipse, JEdit, JBuilder, BlueJ, CodeGuide, NetBeans/Sun Java Studio Enterprise/Creator, IntelliJ IDEA, TextPad, Maven, Ant, Gel, JCreator, and Emacs.

http://pmd.sourceforge.net/

Spøjs note: 2009-02-08 PMD 4.2.5: bug fixes, new rule, new Android ruleset

Hard to do - manual analysis

Hvorfor ikke bare programmere sikkert?

Der er mange ressourcer tilgængelige:

Websites: *Secure Programming for Linux and Unix HOWTO*
http://www.dwheeler.com/secure-programs/

Bøger: *19 Deadly Sins of Software Security: Programming Flaws and How to Fix Them*
Michael Howard, David LeBlanc, John Viega + deres andre bøger

Det er for svært, tager for lang tid!

Feedback

Sørg for feedback i jeres processer

Måske når I kun til denne del, så sørg for at erfaringer opsamles for hvert projekt

Læs ressourcer og lav design så det bliver nemmere at sikre

Få antagelser = færre fejl

Dynamic analysis

compile time vs. at run time nogle fejl kan ikke findes på compile-time

Er du doven så oversæt og kør programmet på OpenBSD ;-)

Part 5 Break it

Use fuzzers, hackertools, improve security by breaking it

Simple fuzzer

```
$ for i in 10 20 30 40 50
>> do
>> ./demo `perl -e "print 'A'x$i"`
>> done
AAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Memory fault
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Memory fault
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Memory fault
```

Memory fault/segmentation fault - juicy!

© copyright 2009 Security6.net, Henrik Lund Kramshøj

Fuzz Revisited

Security .net

Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services

We have tested the reliability of a large collection of basic UNIX utility programs, X-Window applications and servers, and networkservices. We used a simple testing method of subjecting these programs to a random inputstream.

...

The result of our testing is that we can crash (with coredump) or hang (infiniteloop) over 40% (in the worst case) of the basic programs and over 25% of the X-Window applications.

...

We also tested how utility programs checked their return codes from the memory allocation library routines by simulating the unavailability of virtual memory. We could crash almost half of the programs that we tested in this way.

october 1995

Fuzzers

cat /dev/random

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

Et program der kan give forskelligt fejlbehæftet input som måske kan identificere fejl

Jeg anbefaler bogen *Fuzzing: Brute Force Vulnerability Discovery* Michael Sutton, Adam Greene, Pedram Amini og tilhørende website

Se: http://www.fuzzing.org/fuzzing-software

© copyright 2009 Security6.net, Henrik Lund Kramshøj

Fri adgang til hackerværktøjer

I 1993 skrev Dan Farmer og Wietse Venema artiklen
Improving the Security of Your Site by Breaking Into it

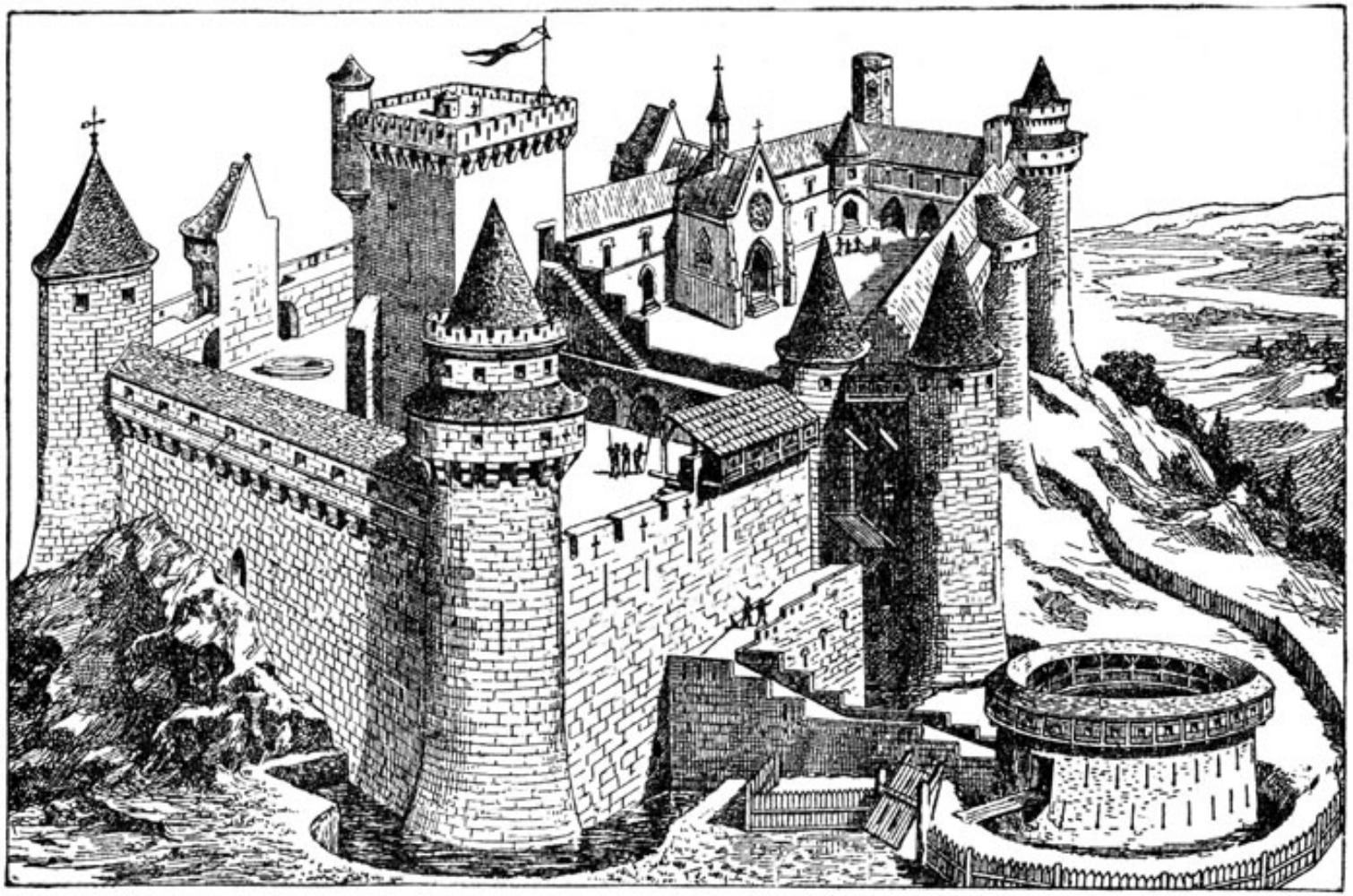
I 1995 udgav de softwarepakken SATAN
Security Administrator Tool for Analyzing Networks

> We realize that SATAN is a two-edged sword - like many tools, it can be used for good and for evil purposes. We also realize that intruders (including wannabees) have much more capable (read intrusive) tools than offered with SATAN.

Traditionen med åbenhed er ført videre helt til idag

Se `http://sectools.org` og `http://www.packetstormsecurity.org/`

Part 6 Enhance and secure runtime environment



Sidste chance er på afviklingstidspunktet

Chroot, Jails and

Der findes mange typer *jails* på Unix

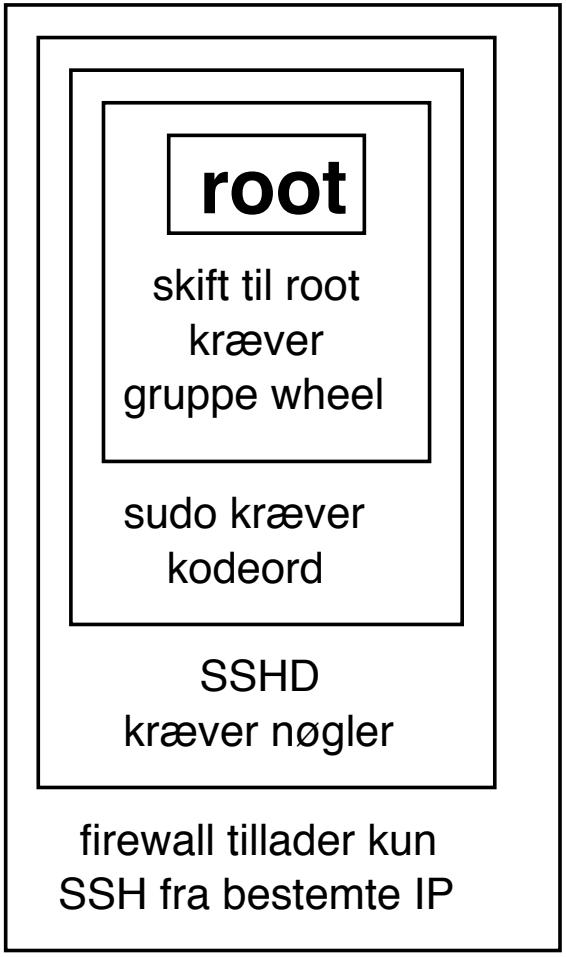
Ideer fra Unix chroot som ikke er en egentlig sikkerhedsfeature

- Unix chroot - bruges stadig, ofte i daemoner som OpenSSH
- FreeBSD Jails
- SELinux
- Solaris Containers og Zones - *jails på steroider*
- VMware virtuelle maskiner, er det et jail?

Hertil kommer et antal andre måder at adskille processer - sandkasser

Husk også de simple, database som `_postgresql`, Tomcat som `tomcat`, Postfix postsystem som `_postfix`, SSHD som `sshd` osv. - simple brugere, få rettigheder

Defense in depth - flere lag af sikkerhed



Forsvar dig selv med flere lag af sikkerhed!

© copyright 2009 Security6.net, Henrik Lund Kramshøj

JVM security policies

Udviklet sammen med Java

Meget kendt

Både Silverlight og JavaFX låner fra denne type model

Apache 6.0.18 catalina.policy (uddrag)

```
// ========== WEB APPLICATION PERMISSIONS ========================================
// These permissions are granted by default to all web applications
// In addition, a web application will be given a read FilePermission
// and JndiPermission for all files and directories in its document root.
grant {
    // Required for JNDI lookup of named JDBC DataSource's and
    // javamail named MimePart DataSource used to send mail
    permission java.util.PropertyPermission "java.home", "read";
    permission java.util.PropertyPermission "java.naming.*", "read";
    permission java.util.PropertyPermission "javax.sql.*", "read";
...
};
// The permission granted to your JDBC driver
// grant codeBase "jar:file:$catalina.home/webapps/examples/WEB-INF/lib/driver.jar!/-" {
//      permission java.net.SocketPermission "dbhost.mycompany.com:5432", "connect";
// };
```

Eksempel fra `apache-tomcat-6.0.18/conf/catalina.policy`

Apple sandbox named generic rules

```
;; named - sandbox profile
;; Copyright (c) 2006-2007 Apple Inc.  All Rights reserved.
;;
;; WARNING: The sandbox rules in this file currently constitute
;; Apple System Private Interface and are subject to change at any time and
;; without notice. The contents of this file are also auto-generated and not
;; user editable; it may be overwritten at any time.
;;
(version 1)
(debug deny)

(import "bsd.sb")

(deny default)
(allow process*)
(deny signal)
(allow sysctl-read)
(allow network*)
```

Apple sandbox named specific rules

```
;; Allow named-specific files
(allow file-write* file-read-data file-read-metadata
  (regex "^(/private)?/var/run/named\\.pid$"
         "^/Library/Logs/named\\.log$"))

(allow file-read-data file-read-metadata
  (regex "^(/private)?/etc/rndc\\.key$"
         "^(/private)?/etc/resolv\\.conf$"
         "^(/private)?/etc/named\\.conf$"
         "^(/private)?/var/named/"))
```

Eksempel fra `/usr/share/sandbox` på Mac OS X

© copyright 2009 Security6.net, Henrik Lund Kramshøj

Gode operativsystemer

Nyere versioner af Microsoft Windows, Mac OS X og Linux distributionerne inkluderer:

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

OpenBSD er nok nået længst og et godt eksempel
http://www.openbsd.org/papers/

NB: meget af dette kræver relativt ny CPU og Memory Management Unit

NB: meget få embedded systemer eller operativsystemer til samme har beskyttelse!

Informationssikkerhed

Husk følgende:

Sikkerhed kommer fra langsigtede intiativer

Hvad er informationssikkerhed?

Data på elektronisk form

Data på fysisk form

Social engineering - *The Art of Deception: Controlling the Human Element of Security* af Kevin D. Mitnick, William L. Simon, Steve Wozniak

Informationssikkerhed er en proces

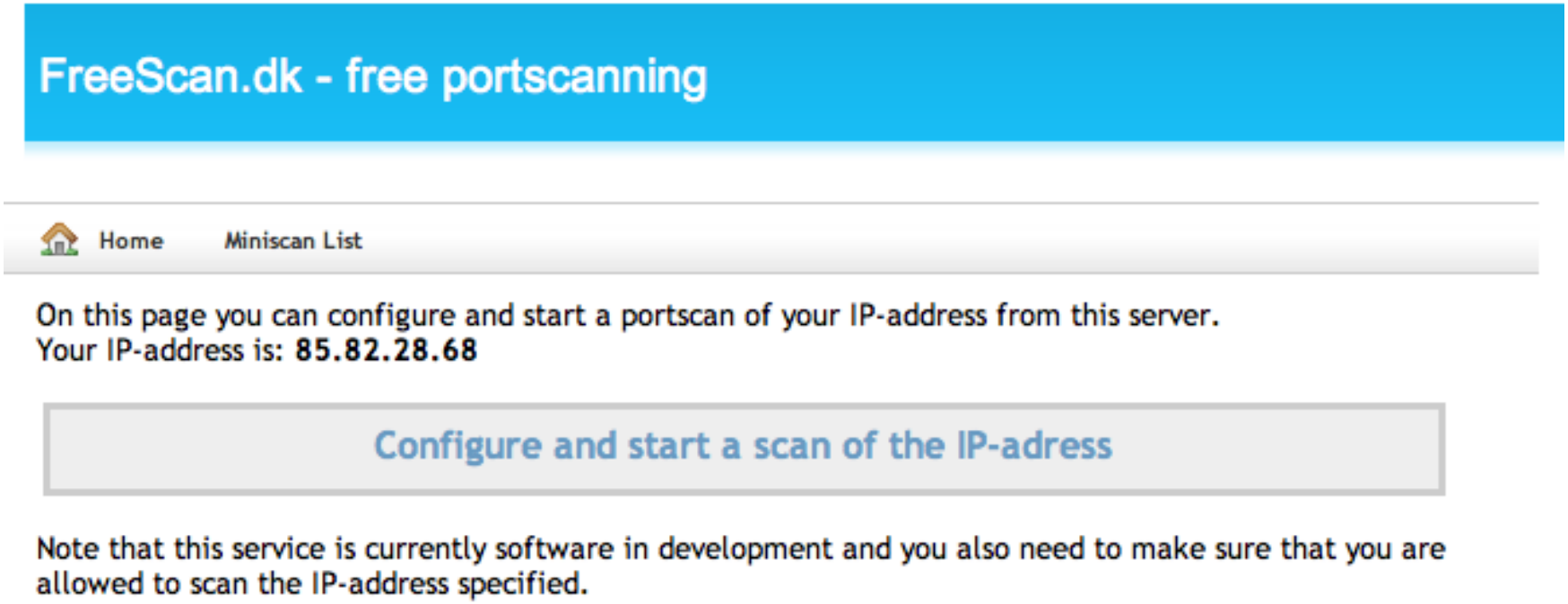
Spørgsmål?

Henrik Lund Kramshøj
hlk@security6.net

http://www.security6.net

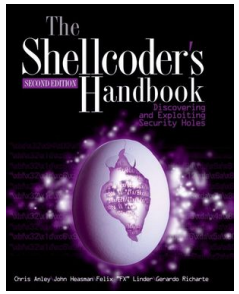
I er altid velkomne til at sende spørgsmål på e-mail

FreeScan.dk - gratis portscanning



http://www.freescan.dk

Buffer overflows



Hvis man vil lære at lave buffer overflows og exploit programmer er følgende dokumenter et godt sted at starte

Smashing The Stack For Fun And Profit Aleph One

Writing Buffer Overflow Exploits with Perl - anno 2000

Følgende bog kan ligeledes anbefales: *The Shellcoder's Handbook : Discovering and Exploiting Security Holes* af Chris Anley, John Heasman, Felix Lindner, Gerardo Richarte 2nd Edition , John Wiley & Sons, august 2007

NB: bogen er avanceret og således IKKE for begyndere!

milw0rm - dagens buffer overflow

[home] [contents] [platforms] [shellcode] [search] [cracker] [links] [rss] [archive]

MILW0RM

[highlighted]

-::DATE	-::DESCRIPTION	-::HITS					-::AUTHOR
2009-03-05	Winamp <= 5.541 Skin Universal Buffer Overflow Exploit	3128	R		D		SkD
2009-02-26	Coppermine Photo Gallery <= 1.4.20 (BBCode IMG) Privilege Escalation	7338	R		D		StAkeR
2009-02-25	Apple MACOS X xnu <= 1228.x Local Kernel Memory Disclosure Exploit	4111	R		D		mu-b
2009-02-23	Adobe Acrobat Reader JBIG2 Local Buffer Overflow PoC #2 0day	17652	R		D		Guido Landi
2009-02-23	MLdonkey <= 2.9.7 HTTP DOUBLE SLASH Arbitrary File Disclosure Vuln	4225	R		D		Michael Peselnik
2009-02-23	Multiple PDF Readers JBIG2 Local Buffer Overflow PoC	7781	R		D		webDEVIL

[remote]

-::DATE	-::DESCRIPTION	-::HITS					-::AUTHOR
2009-03-05	SupportSoft DNA Editor Module (dnaedit.dll) Code Execution Exploit	1093	R		D	X	Nine:Situations:Group
2009-03-04	Easy File Sharing Web Server 4.8 File Disclosure Vulnerability	1424	R		D		Stack
2009-03-04	EFS Easy Chat Server Authentication Request Buffer Overflow Exploit (pl)	969	R		D		Dr4sH
2009-03-04	MS Internet Explorer 7 Memory Corruption Exploit (MS09-002) (fast)	3965	R		D		Ahmed Obied
2009-03-03	EFS Easy Chat Server (XSRF) Change Admin Pass Vulnerability	1215	R		D		Stack
2009-03-03	Imera ImeraIEPlugin ActiveX Control Remote Code Execution Exploit	1020	R		D		Elazar

[local]

-::DATE	-::DESCRIPTION	-::HITS					-::AUTHOR
2009-03-05	Media Commands (m3u File) Universal SEH Overwrite Exploit	669	R		D		His0k4
2009-03-05	Media Commands .m3l File Local Buffer Overflow Exploit	621	R		D		Stack

http://milw0rm.com/

Metasploit

What is it?

The Metasploit Framework is a development platform for creating security tools and exploits. The framework is used by network security professionals to perform penetration tests, system administrators to verify patch installations, product vendors to perform regression testing, and security researchers world-wide. The framework is written in the Ruby programming language and includes components written in C and assembler.

Trinity brugte et exploit program ☺

Idag findes der samlinger af exploits som milw0rm

Udviklingsværktøjerne til exploits er idag meget raffinerede!

http://www.metasploit.com/

Reklamer: kursusafholdelse

Security6.net

Følgende kurser afholdes med mig som underviser

- IPv6 workshop - 1 dag
  Introduktion til Internetprotokollerne og forberedelse til implementering i egne netværk.
- Wireless teknologier og sikkerhed workshop - 2 dage
  En dag med fokus på netværksdesign og fornuftig implementation af trådløse netværk, samt integration med hjemmepc og wirksomhedsnetværk.
- Hacker workshop 2 dage
  Workshop med detaljeret gennemgang af hackermetoderne angreb over netværk, exploitprogrammer, portscanning, Nessus m.fl.
- Forensics workshop 2 dage
  Med fokus på tilgængelige open source værktøjer gennemgås metoder og praksis af undersøgelse af diskimages og spor på computer systemer
- Moderne Firewalls og Internetsikkerhed 2 dage
  Informere om trusler og aktivitet på Internet, samt give et bud på hvorledes en avanceret moderne firewall idag kunne konfigureres.

Se mere på `http://www.security6.net/courses.html`

© copyright 2009 Security6.net, Henrik Lund Kramshøj