

Introduction to GNU/Linux hardening



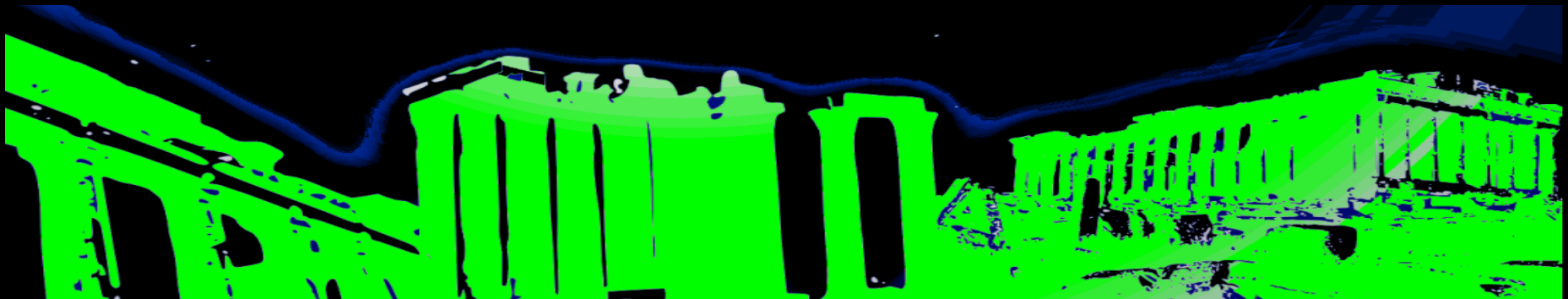
OpenSUSE conference 2013

Thessaloniki, Greece

Γεια σου, Θεσσαλονίκη!

Shawn the R0ck

July 21 2013



cat /proc/agenda

- * Who am I?
- * Smash the stack in old good hacking days
- * Easy ways to Hardening
 - * Mitigation tech via GCC/glibc
 - * Network
 - * Confinement
- * What is security
- * Why exploit
 - * for Fun and... Profit
 - * Hacker ethic



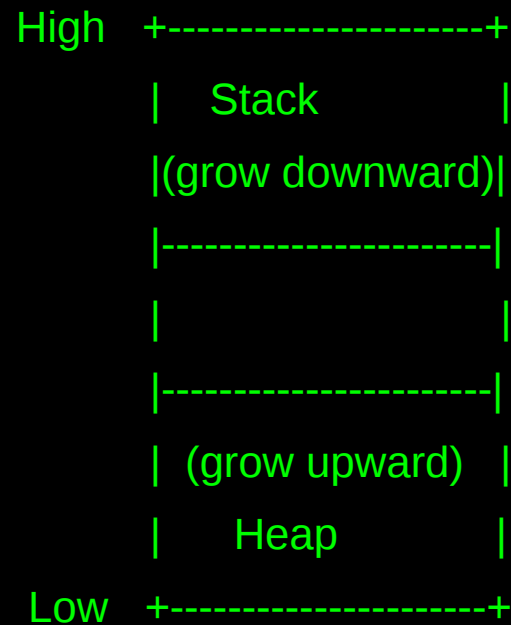
whois

- * Chinese dude
 - * IT
 - * Security guy
 - * Monkey coder
 - * Day job at SUSE Linux



Smashing the stack

- * Stack/Heap layout
- * Stack grows down (x86, MIPS)
- * ESP points to the current top of the stack
- * EBP points to the current function frame



Example of overwrite ret

```
#include <stdio.h>
#include <string.h>
```

```
void test(const char *input)
{
    int x; /* integer variable need 4-bytes in memory */
    char buf[10];
    strcpy(buf, input);
    printf("%s \n", buf);
}
```



```
int main(int argc, char *argv[])
{
    printf("The address of func test(): %p, func fuck_me(): %p\n", test, fuck_me);
    test(argv[1]);  -----> It's a vulnerable func without bound checks!
    return 0;
}
```

Further reading/hacking

- * Smashing the Stack for Fun and Profit, by Aleph One, Phrack Issue 49

- * Join the io-wargame:

<http://smashthestack.org/index.php>

- * Open Security Training:

<http://www.opensecuritytraining.info/Welcome.html>

Mitigation tech via GCC/glibc

- * Stack canary

--> A “canary” value is placed between frame pointer and data on the stack

- * Position-Independent-Executable

--> PIE enforces every process's code segment is mmap()'d, it begins at a different base address at each execution of the application.

- * Other



Stack canary

GCC options:

-fstack-protector, -fstack-protector-all

Example?

```
Int fuc_me(int x, int y) /* x? WTH */
{
    Int v; /* v? */
    char buf[256];
    int h;

    .....
}
```

Original layout:

[High addr...[y].[x].[ret].[frame pointer].[v].[buf].[h]...Low addr]

Simple Canary layout:

[High addr...[y].[x].[ret].[frame pointer].[canary].[v].[buf].[h]...Low addr]

But, it might looks like this:

[High addr...[y].[x].[ret].[frame pointer].[carnary].[buf].[h].[v]...Low addr]



Position-Independent-Executable

GCC options:

-pie, it would randomize the .text segment

Example:

```
void* getEIP () {  
    return __builtin_return_address(0);  
};
```

```
int main(int argc, char** argv){  
    printf("retaddr: %p\n",getEIP());  
    return 0;  
}
```

retaddr: 0xb78d950a
retaddr: 0xb77a450a
retaddr: 0xb776350a

Turn off ASLR

```
root@sl13:/tmp# echo 0 > /proc/sys/kernel/randomize_va_space  
root@sl13:/tmp# ./a.out  
retaddr: 0xb7fff50a  
root@sl13:/tmp# ./a.out  
retaddr: 0xb7fff50a  
root@sl13:/tmp# ./a.out  
retaddr: 0xb7fff50a  
root@sl13:/tmp# echo 2 > /proc/sys/kernel/randomize_va_space  
root@sl13:/tmp# ./a.out  
retaddr: 0xb77d450a  
root@sl13:/tmp# ./a.out  
retaddr: 0xb780c50a  
root@sl13:/tmp# ./a.out  
retaddr: 0xb786850a  
root@sl13:/tmp#
```

Turn on ASLR

Other mitigation in GCC

GCC options:

-FORTIFY_SOURCE, prevent string format vulnerability

-z nostack, non-executable stack

-z relro, -z relro -z now, read-only relocation – it's a hardening option for GOT

Bypass these mitigations

Reference:

Scraps of notes on remote stack overflow exploitation, Phrack Issue 67

<http://www.phrack.org/issues.html?issue=67&id=13&mode=txt>

Bypassing PaX ASLR protection, Phrack Issue 59

<http://phrack.org/issues.html?issue=59&id=9&mode=txt>

The Art Of ELF: Analysis and Exploitations

<http://fluxius.handgrep.se/2011/10/20/the-art-of-elf-analysises-and-exp>

A Eulogy for Format Strings

<http://www.phrack.org/issues.html?issue=67&id=9&mode=txt>

Network

Network policy:

Tcp wrapper, it's userspace implementation of host-based networking ACL. It's easy to use:

/etc/hosts.allow, /etc/hosts.deny



iptables/netfilter, SuSEFirewall or your own policies? It's only 5-tuple matters.

Do you really need IDS(Snort, BRO) or layer-7 classifier(I7, opendpi)?



Confinement

Why do we need SELinux or Apparmor? 0Day attack mitigation?

Apparmor, whitelist-based policy. Example:

```
/home/shawn/a.out {  
    #include <abstractions/base>
```

```
  
    /home/shawn/a.out mr,  
    /home/shawn/hello r,  
    /home/shawn/world w,  
    network stream,  
}
```



cat /proc/security

Security is NOT:

- * Security is NOT installing a firewall
- * Security is NOT a Product or Service
- * Security is Not a Product; It's a Process

Security is:

- * Security is a Process, Methodology, Costs, Policies and People
- * Security is only as good as your "weakest link"
- * Security is 24x7x365 ... constantly ongoing .. never ending

Conclusion:

- * Hardening should be part of your security process

Why exploit

- * Motivation

- * Fun? To some hackers, hacking is part of their life. They can't live without hacking. They are happy with joy while writing exploit...

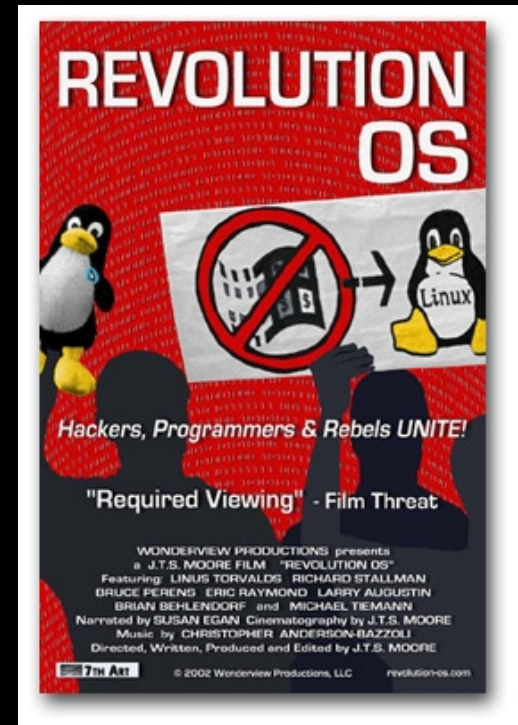
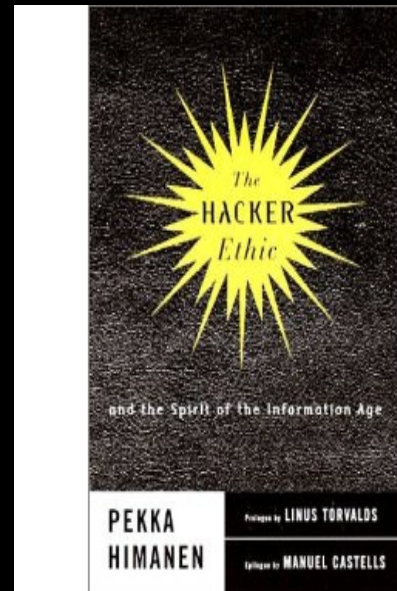
- * Profit? Money, of course. White hat working for commercial company. Black hat?

- * Both White and Black are possible to sale exploit in underground.



Hacker ethic?

- * For further reading/hacking.
- * Book
 - * The HACKER Ethic and the Spirit of Information Age
- * Documentary film
 - * Revolution OS
- * The best security ezine
 - * Phrack



PHRACK

Join the Geeko!

Learn more about SUSE's
openings, globally:

- Talk to our colleagues at the booth
- Check out our careers page www.suse.com/careers
- Contact our recruiting team at jobs@suse.com



Questions?

Download examples:

- <https://github.com/citypw/citypw-SCFE/tree/master>

Thanks!

Drop me a line if you want:

<Shawn the R0ck, citypw@gmail.com>

Reference

SUSE Linux Enterprise Server 11 SP3
Security and Hardening:

<https://www.suse.com/documentation/sles11/single>

Phrack:

<http://phrack.org/>