A Deeper Perspective

OpenBSD

Karthik Kumar Viswanathan

An Introduction

- OpenBSD: Free, Functional, Secure
- Originally based on BSD 4.4 UNIX
- Emphasis on correctness, standardization, portability and most importantly, **security**
- Founded by Theo de Raadt in 1995
- Latest version is **4.0**
- Available on: AMD64, i386, Sparc64, Sparc, PPC, ARM, Alpha and others

Why OpenBSD?

- When your goal is setting up a secure server
- When your goal is setting up a great firewall
- When your goal is setting up a secure machine for limited desktop use

Why not OpenBSD?

- When your goal is to use it for extreme graphics, gaming and the like
- When your goal is to make use of as many blob drivers to achieve the previous goal
- When your goal is to look at it as a replacement of Linux for the previous goal
- When your goal is to use less secure protocols like the WPA and anything that supports it, forget it

Overview of Security Features

- Written with security in mind
- Simplicity that works
- Memory protection
- Cryptography
- Privilege Separation and Revocation
- Packet Filter
- Miscellaneous
- Patch, Patch

Written with security in mind

- Security is enforced by default
- The default install has had *two* remote vulnerabilities in the last ten years (OpenSSHv1 and ICMPv6-mbuf)
- Built in Cryptography support at lowest levels
- Built in Memory protection and execution prevention and lowest levels
- Full disclosure of vulnerabilities
- Source code audits
- Fast fix-arounds for vulnerabilities
- Continuous improvement

Simplicity that works

- Don't keep things which can break: **telnetd**
- Keep sets of parameters which have proven to be computationally secure: Secure by default
- Passwords use Blowfish **bcrypt**
- IPSec is easy to set up, uses HMAC-SHA1/2, AES, MODP1024
- PF is a very simple but powerful packet filter

Memory protection

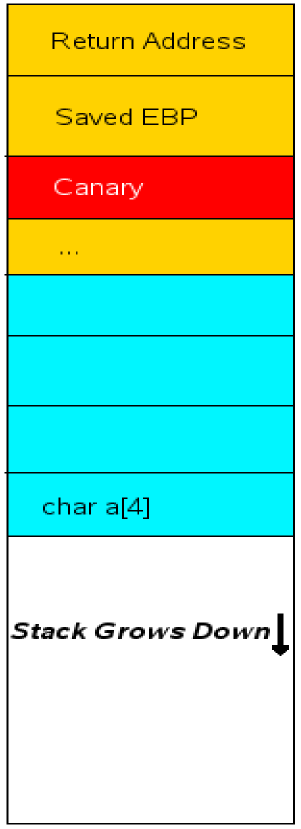
- Changes to *malloc* routines to use *mmap* to not allocate segments contiguously
- *strlcpy()* and *strlcat()*
- Pure .rodata segment in code segment
- Propolice: Detect stack smashes and don't execute things you aren't sure of
- W^X Protection: Either write or execute it
- ASLR: Randomize locations of important system routines, especially exec

Propolice

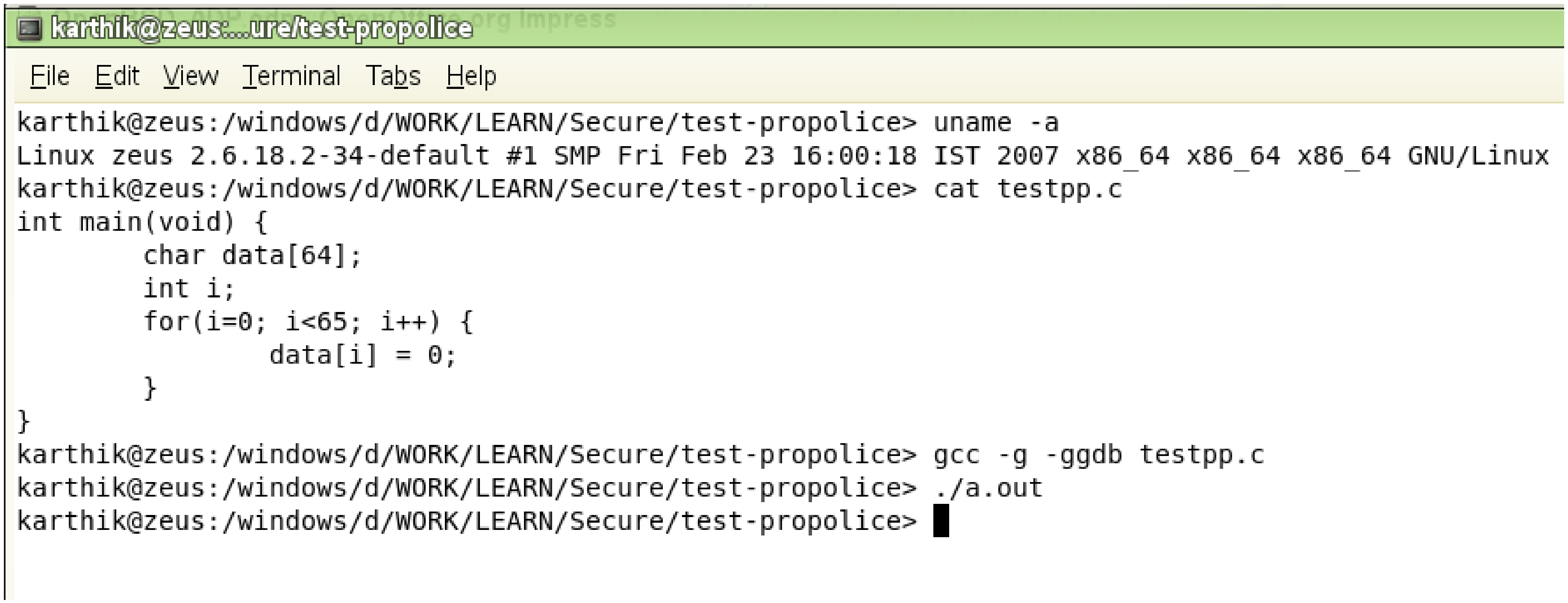
- Detects stack smashes
- Uses a canary on the stack, after Return Pointer and EBP and before local variables
- Changing the return pointer will imply overwriting the canary.
- Is detected by code before return

Propolice

```
        je      .L3
        call    __stack_chk_fail
.L3:
        leave
        ret
```

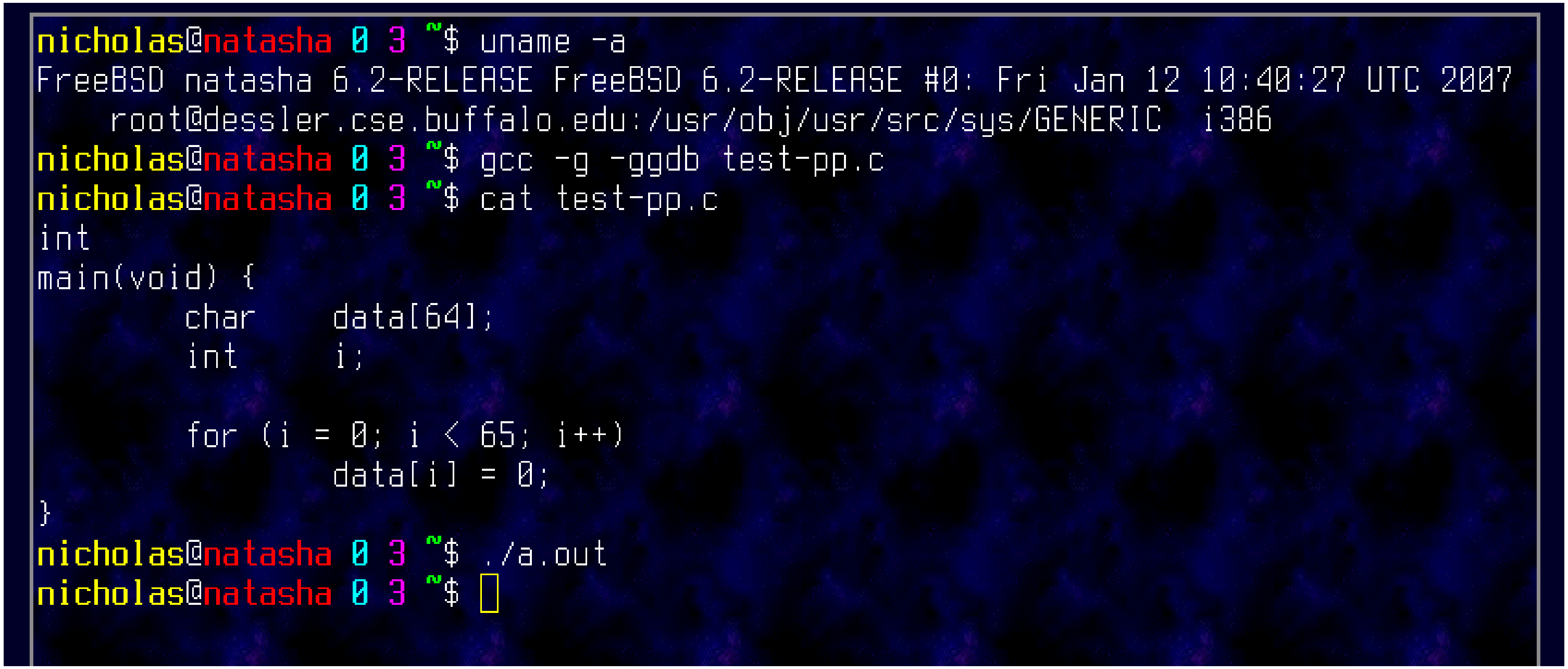


on Linux


karthik@zeus:...ure/test-propolice
File Edit View Terminal Tabs Help

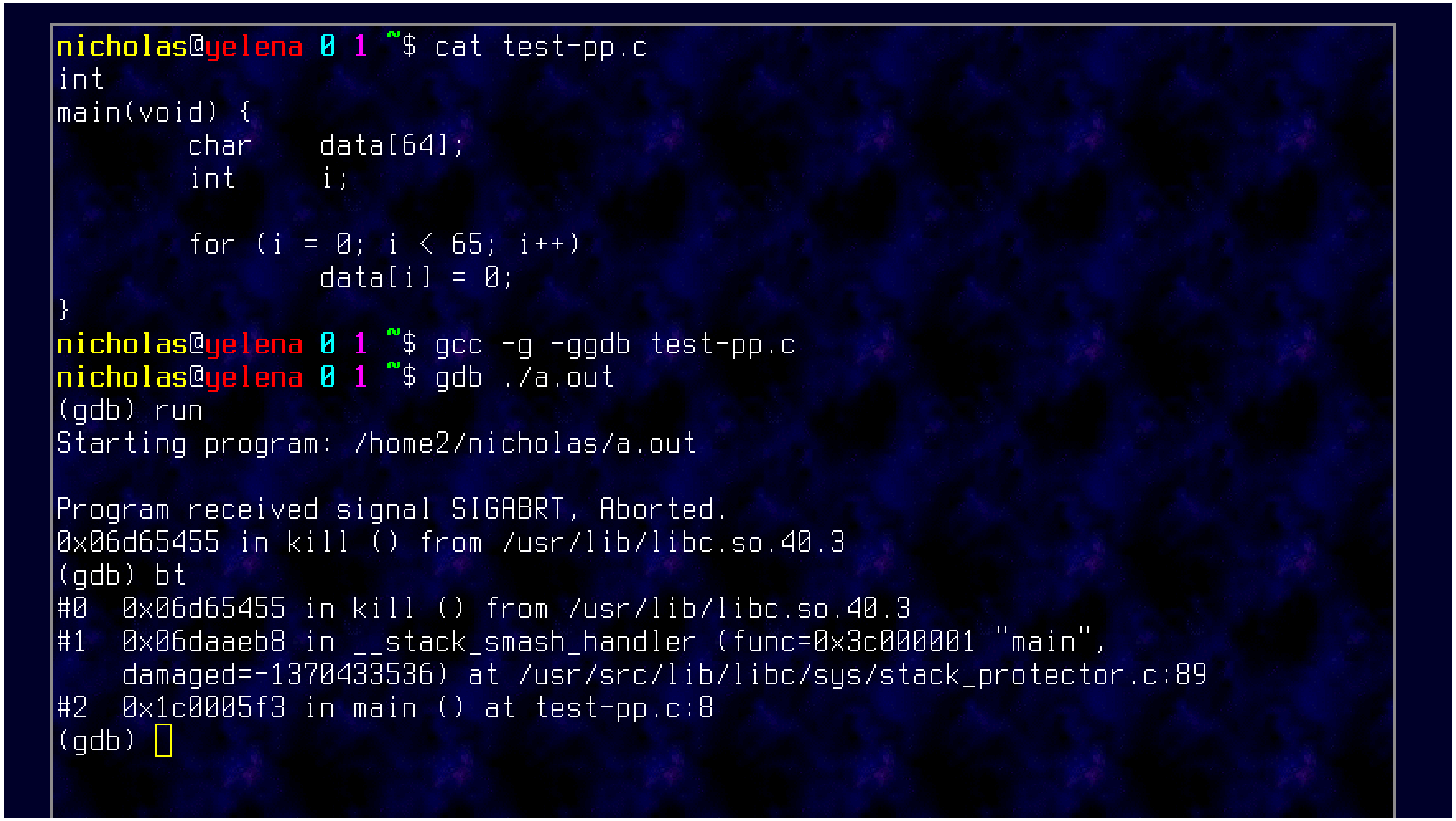
```
karthik@zeus:/windows/d/WORK/LEARN/Secure/test-propolice> uname -a
Linux zeus 2.6.18.2-34-default #1 SMP Fri Feb 23 16:00:18 IST 2007 x86_64 x86_64 x86_64 GNU/Linux
karthik@zeus:/windows/d/WORK/LEARN/Secure/test-propolice> cat testpp.c
int main(void) {
        char data[64];
        int i;
        for(i=0; i<65; i++) {
                data[i] = 0;
        }
}
karthik@zeus:/windows/d/WORK/LEARN/Secure/test-propolice> gcc -g -ggdb testpp.c
karthik@zeus:/windows/d/WORK/LEARN/Secure/test-propolice> ./a.out
karthik@zeus:/windows/d/WORK/LEARN/Secure/test-propolice> █
```

on FreeBSD



```
nicholas@natasha 0 3 ~$ uname -a
FreeBSD natasha 6.2-RELEASE FreeBSD 6.2-RELEASE #0: Fri Jan 12 10:40:27 UTC 2007
    root@dessler.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC  i386
nicholas@natasha 0 3 ~$ gcc -g -ggdb test-pp.c
nicholas@natasha 0 3 ~$ cat test-pp.c
int
main(void) {
        char    data[64];
        int     i;

        for (i = 0; i < 65; i++)
                data[i] = 0;
}
nicholas@natasha 0 3 ~$ ./a.out
nicholas@natasha 0 3 ~$ 
```

on OpenBSD



```
nicholas@yelena 0 1 ~$ cat test-pp.c
int
main(void) {
        char    data[64];
        int     i;

        for (i = 0; i < 65; i++)
                data[i] = 0;
}
nicholas@yelena 0 1 ~$ gcc -g -ggdb test-pp.c
nicholas@yelena 0 1 ~$ gdb ./a.out
(gdb) run
Starting program: /home2/nicholas/a.out

Program received signal SIGABRT, Aborted.
0x06d65455 in kill () from /usr/lib/libc.so.40.3
(gdb) bt
#0  0x06d65455 in kill () from /usr/lib/libc.so.40.3
#1  0x06daaeb8 in __stack_smash_handler (func=0x3c000001 "main",
    damaged=-1370433536) at /usr/src/lib/libc/sys/stack_protector.c:89
#2  0x1c0005f3 in main () at test-pp.c:8
(gdb) 
```

OpenBSD

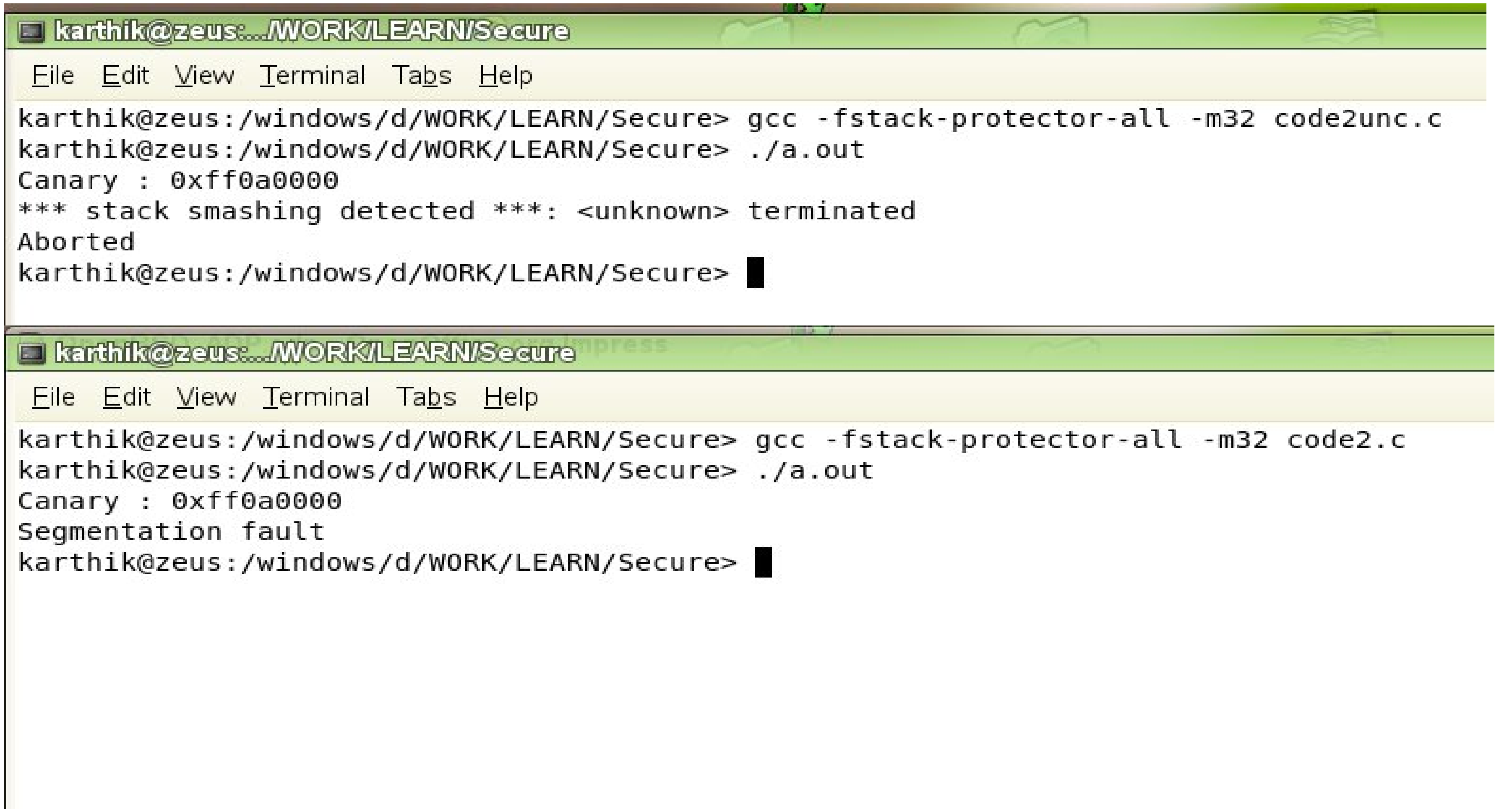
Propolice

- Can be enabled explicitly by `-fstack-protector-all` on Linux and FreeBSD
- Used to be an extension, now part of the GCC Compiler 4.1
- Secure, but not secure enough

Anti-Propolice

```
//Should work on x86, gcc 4.1.x.
//Must compile with -fstack-protector-all -m32
#include <stdio.h>

int canary=0;

int detectcanary(){
        int a[4];
        canary=a[4];
        return 1;
}

int overridedata() {
        int i;
        int data[1];

        printf("Canary : 0x%07x\n",canary);

        for(i=0; i<128; i++) {
            // Protect Overwriting canary
            if (data[i]==canary) continue;
            data[i] = 0;
      }
}

int main()
{
        int c,d;
        c=detectcanary();
        d=overridedata();
        return 0;
}
```

on Linux


karthik@zeus:.../WORK/LEARN/Secure
File Edit View Terminal Tabs Help

```
karthik@zeus:/windows/d/WORK/LEARN/Secure> gcc -fstack-protector-all -m32 code2unc.c
karthik@zeus:/windows/d/WORK/LEARN/Secure> ./a.out
Canary : 0xff0a0000
*** stack smashing detected ***: <unknown> terminated
Aborted
karthik@zeus:/windows/d/WORK/LEARN/Secure>
```

karthik@zeus:.../WORK/LEARN/Secure
File Edit View Terminal Tabs Help

```
karthik@zeus:/windows/d/WORK/LEARN/Secure> gcc -fstack-protector-all -m32 code2.c
karthik@zeus:/windows/d/WORK/LEARN/Secure> ./a.out
Canary : 0xff0a0000
Segmentation fault
karthik@zeus:/windows/d/WORK/LEARN/Secure>
```

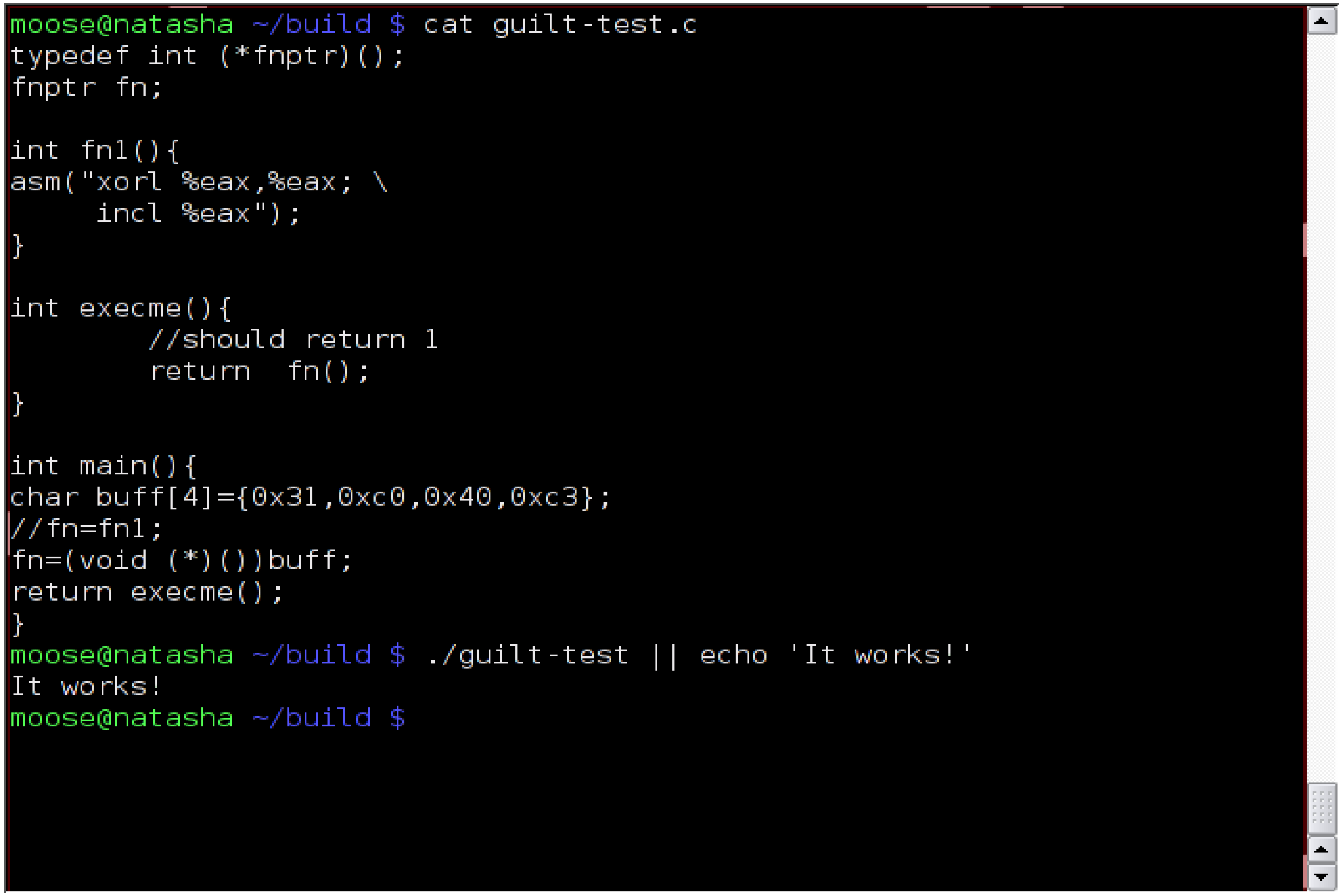
W^X Protection

- One step further into memory protection
- Either Write or Execute, not both
- Execute disable bit (NX) not available in i386 platform
- Has to be emulated
- A good feature which prevents execution off stack
- Heap execution may be a bad idea sometimes

W^X Protection

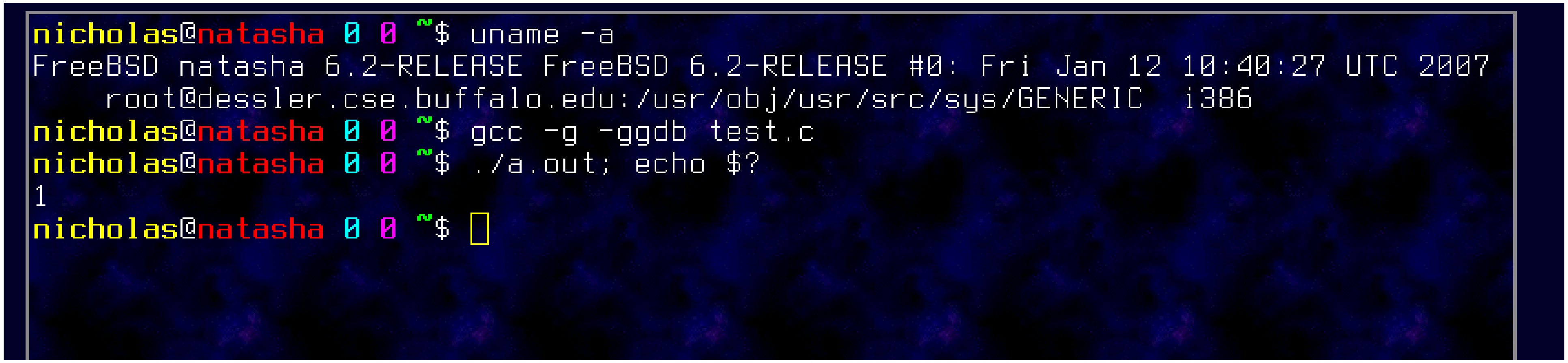
```
//Should work on executable segments. Especially stack and/or heap

typedef int (*fnptr)();
fnptr fn;

int execme(){
        //should return 1
        return  fn();
}

int main(){
char buff[4]={0x31,0xc0,0x40,0xc3};
fn=(void *)buff;
return execme();
}
```

On Linux without SELinux



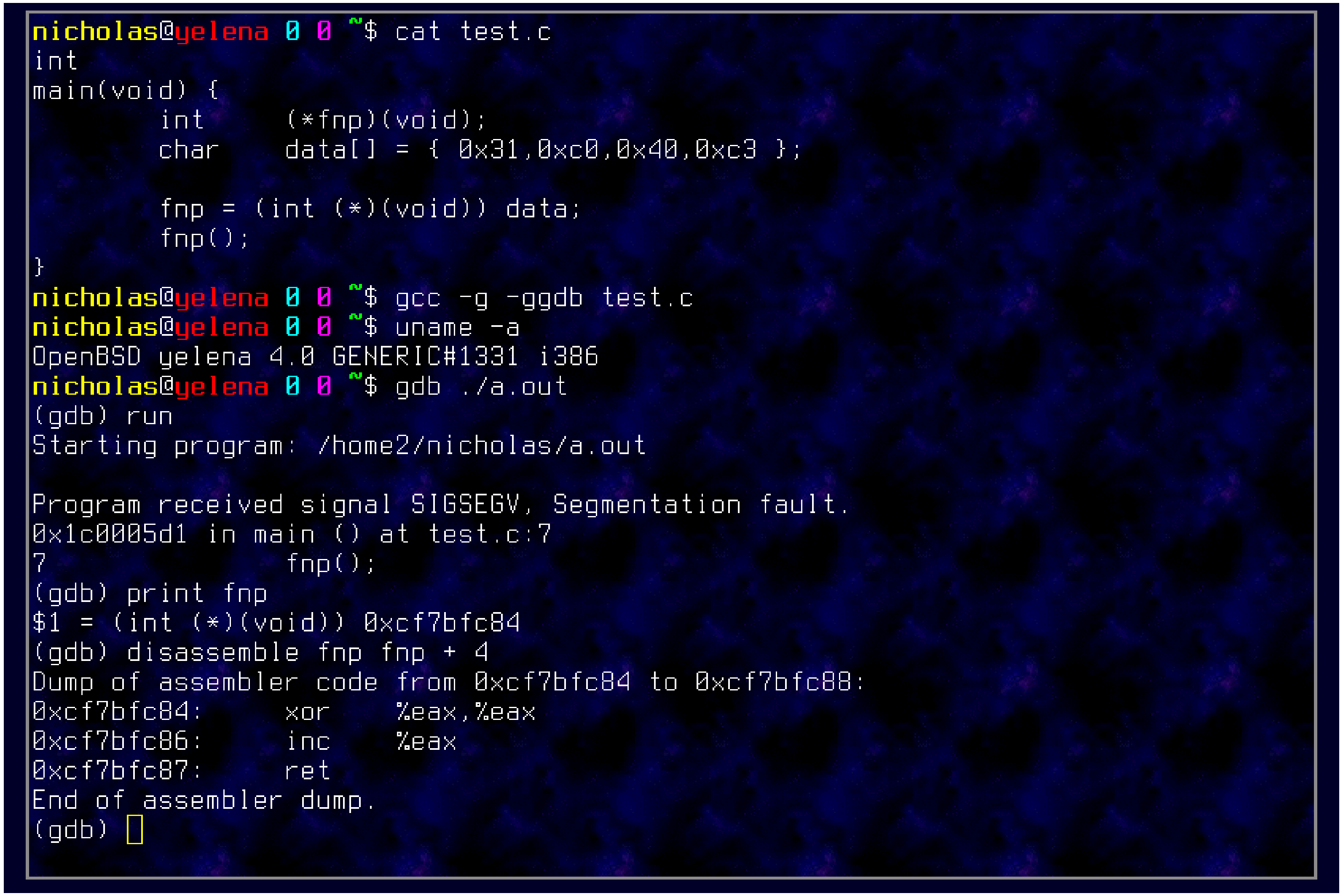
```
moose@natasha ~/build $ cat guilt-test.c
typedef int (*fnptr)();
fnptr fn;

int fn1(){
asm("xorl %eax,%eax; \
     incl %eax");
}

int execme(){
        //should return 1
        return  fn();
}

int main(){
char buff[4]={0x31,0xc0,0x40,0xc3};
//fn=fn1;
fn=(void (*)())buff;
return execme();
}
moose@natasha ~/build $ ./guilt-test || echo 'It works!'
It works!
moose@natasha ~/build $
```

On FreeBSD



```
nicholas@natasha 0 0 ~$ uname -a
FreeBSD natasha 6.2-RELEASE FreeBSD 6.2-RELEASE #0: Fri Jan 12 10:40:27 UTC 2007
    root@dessler.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC  i386
nicholas@natasha 0 0 ~$ gcc -g -ggdb test.c
nicholas@natasha 0 0 ~$ ./a.out; echo $?
1
nicholas@natasha 0 0 ~$ 
```

On OpenBSD



```
nicholas@yelena 0 0 ~$ cat test.c
int
main(void) {
        int     (*fnp)(void);
        char    data[] = { 0x31,0xc0,0x40,0xc3 };

        fnp = (int (*)(void)) data;
        fnp();
}
nicholas@yelena 0 0 ~$ gcc -g -ggdb test.c
nicholas@yelena 0 0 ~$ uname -a
OpenBSD yelena 4.0 GENERIC#1331 i386
nicholas@yelena 0 0 ~$ gdb ./a.out
(gdb) run
Starting program: /home2/nicholas/a.out

Program received signal SIGSEGV, Segmentation fault.
0x1c0005d1 in main () at test.c:7
7               fnp();
(gdb) print fnp
$1 = (int (*)(void)) 0xcf7bfc84
(gdb) disassemble fnp fnp + 4
Dump of assembler code from 0xcf7bfc84 to 0xcf7bfc88:
0xcf7bfc84:     xor     %eax,%eax
0xcf7bfc86:     inc     %eax
0xcf7bfc87:     ret
End of assembler dump.
(gdb)
```

ASLR

- One step further into memory protection
- Randomizes addresses of standard routines to prevent return to addresses of standard functions like *exec()*, *fork()*
- Reduces the damage done greatly by buffer overflows

Cryptography

- Strong PRNGs
- Built in cryptographic hash functions and support for cryptographic hardware
- Passwords use Blowfish: CPU intensive and makes brute force attackers' day longer and harder
- Encryption of multiple portions of the swap partition using different keys

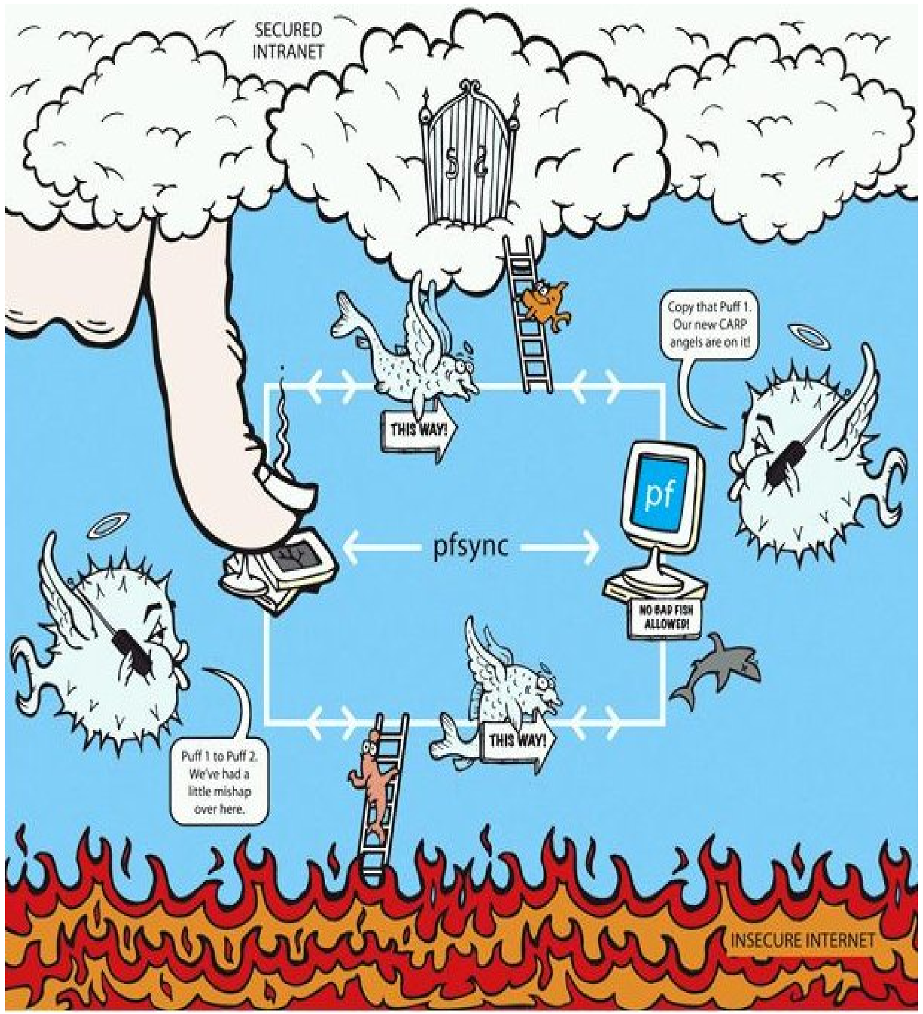
Cryptography

- Network stack makes heavy use of the randomization provided for *nonces*, *TCP sequence numbers* and *ephemeral client port numbers*.
- Includes *IPSec* and *Kerberos*

Privilege Separation and Revocation

- Run daemons at lowest possible privileges
- Is enough 99% of the time
- Don't grant uid0 privileges unless absolutely required
- Revoke privileges once uid0 operations are performed
- Protects against many known access vulnerabilities

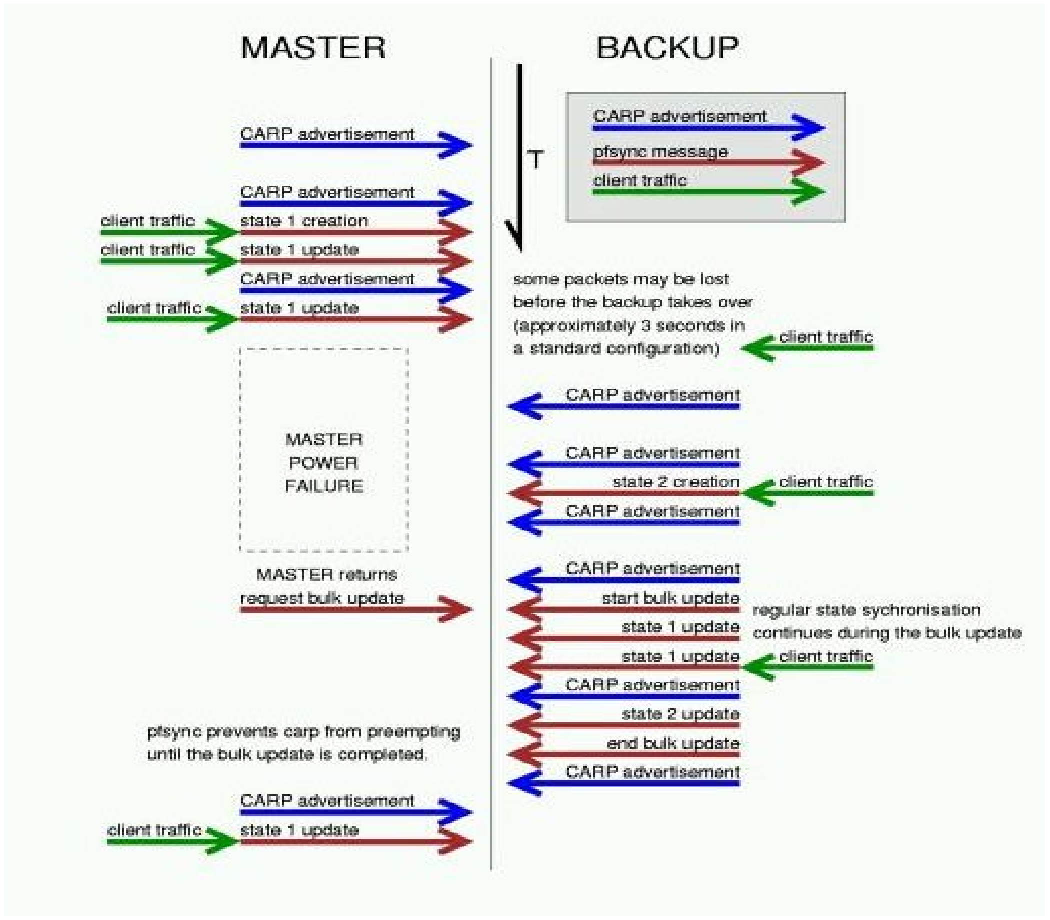
Packet Filter


SECURED INTRANET
Copy that Puff 1. Our new CARP angels are on it!
THIS WAY!
pfsync
pf
NO BAD FISH ALLOWED!
Puff 1 to Puff 2. We've had a little mishap over here.
THIS WAY!
INSECURE INTERNET

Packet Filter

- Normalization (sanitize packets)
- Filtering (including TCP attributes)
- Translation (NAT, RDR, BINAT)
- DoS Mitigation (*max-src-nodes*, congestion handling, bandwidth throttling)
- Redundancy: *pfsync* and *CARP*
- *pfsync* synchronizes rules
- *CARP* provides failovers to hosts and routers
- *CARP* uses virtual MAC addresses and ARP

Synchronization



Miscellaneous

- *chroot jails* for Apache and other important daemons
- atexit() process list and stdio protection
- OpenBSD has very few security vulnerabilities compared to other distributions
- Great suite: OpenSSH, OpenCVS, OpenBGPD, OpenNTPD

Patch, Patch

- OpenBSD has *sendmail* which is usually considered very insecure
- OpenBSD has a heavily patched *httpd* 1.3.x
- OpenBSD has very few security vulnerabilities compared to other OSes/distributions
- Keep patching with OpenBSD's security advisories

Upcoming Release

- **4.1**, May 1 2007
- Better ACPI Support
- Better Sparc64 Support
- *hoststated* for Layer 3/7 Load balancing
- Many drivers, improvements and bug fixes!

MULTI-PLATFORM ULTRA SECURE OPERATING SYSTEM
OpenBSD4.1
CD1: i386
CD2: macppc • amd64 • track 2 is audio track
CD3: sparc • sparc64 • source code
CD can boot on all architectures. Other architectures only available via FTP

Questions?

?????????????????????

???????????????????

??????????????????????

???????????????????

?????????????????????

?????????????????????

???????????????????????

??????????????????????

??????????????

??????????????

??????????????

??????????????

Links

- http://openbsd.org
- http://undeadly.org
- http://www.openfaqs.org/
- http://www.openbsd101.com

References

- The OpenBSD FAQ
- OpenBSD 101
- http://cvs.openbsd.org/papers/bsdcan04-pf/
- http://en.wikipedia.org/wiki/OpenBSD_security_features

Acknowledgments

My heartfelt thanks go to:

Network Security Forum Members

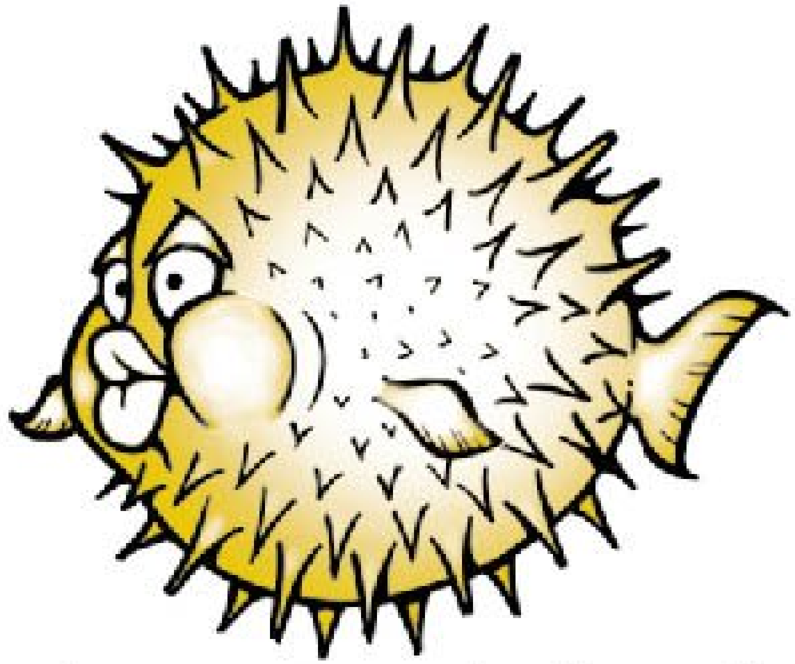
Murali P

Subba Reddy

Freenode #allegro: *Tomasu*, *CGamesPlay*

Freenode #openbsd: *NicM*, *bofh*

Freenode #freebsd: *zcram*



So long, and thanks for all the passwords