Security - Malware

Note Title 2/14/2011

Announcements

- Lab 2 is due in 1 week.
  (You should have already started!)

- Next paper summary will be due next Thursday.

Viruses, Worms, & root kits

Computer attacks have become a reality in the modern world.

Business & individuals lose millions to bugs that are preventable.

But what is really involved?

Definitions

name	description
virus	replicates itself into other executable code; when the infected code is executed, the virus also executes
worm	runs independently; propagate a complete working version of itself onto other hosts.
logic bomb	inserted into software by an attacker; lies dormant until some condition is met.
Trojan horse	appears to have a useful function, but also has a hidden, malicious function.
backdoor	any mechanism that bypasses a normal security check; sometimes intentional, sometimes accidental and exploited.
auto-rooter	a set of scripts used to break into machines remotely.
rootkit	a set of scripts used maintain admin access on a machine.

Taken From "Computer Security" by Stallings and Brown, Chapter 7, Table 7.1

Reality

Writing any of these requires a lot of system programming knowledge!

(Don't be a script kiddie!)

Virus

- Generally embedded in another piece of code (often compressed to avoid detection)

Many ways to classify them:

classification	description
boot sector infector	infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
file infector	infects files that the operating system or shell consider to be executable.
macro virus	infects files with macro code that is interpreted by an application.

Viruses generally hide somehow:

classification	description
encrypted virus	a portion of the virus creates an encryption key and encrypts the remainder of the virus; because the virus is encrypted, it's difficult to scan.
stealth virus	explicitly hides itself from detection; the entire virus is hidden.
polymorphic virus	mutates with every infection, making signature scan difficult.
metamorphic virus	rewrites itself with every infection, making signature scan difficult.

Worms

In contrast, worms generally run independently and propogate themselves via network connections

Today, we'll examine one particular worm which exploits buffer overflow vulnerabilities.

In general, worms often exploit some vulnerability in a program that is already present on the system.

Early Example: Sendmail

- UNIX mail server. Early versions had 2 features

(1) An outside SMTP connection could place the program into DEBUG mode

(2) In DEBUG mode, a user could execute the "run command" feature, which sent an email to a program. The program would use the contents of the email as input.

Another: Finger

A UNIX command which provides user information:

```
$ finger oster
Login: oster Name: Osterberg
Directory: /Users/oster Shell: /
bin/bash
Office: Peter
Never logged in.
No Mail.
No Plan
```

Problem:

Finger uses the function gets() from the C library.

Problem: Takes input from command prompt without a maximum length.

In C, "strings" are arrays with fixed size!

What is bounds checking?

Consider this C program:

```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

only 16 characters

What does it do?

Just prompts for input & then prints it.

```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

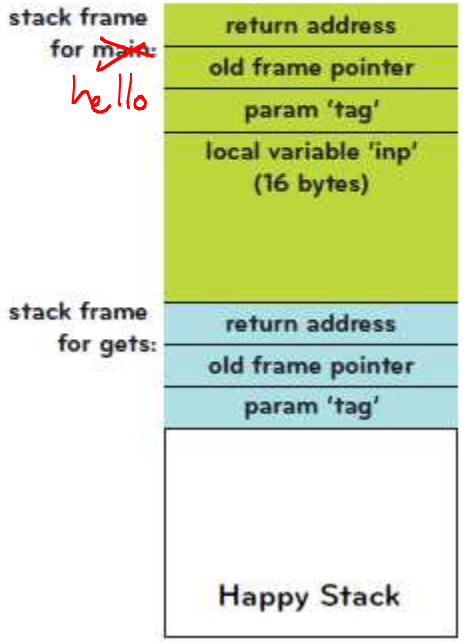
```
$ cc -g -o buffer2 buffer2.c

$ ./buffer2
Enter value for name: Bill and Lawrie
Hello your name is Bill and Lawrie
buffer2 done

$ ./buffer2
Enter value for name:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Segmentation fault (core dumped)

$ perl -e 'print pack("H*",
"41424344454647485152535455565758616263646566676808
fcffbf948304080a4e4e4e4e0a");' | ./buffer2
Enter value for name:
Hello your Re?pyy|uEA is ABCDEFGHQRSTUVWXabcdefquyu
Enter value for Kyyu:
Hello your Kyyu is NNNN
Segmentation fault (core dumped)
```

Taken From "Computer Security" by Stallings and Brown, Figure 11.5

What is happening?

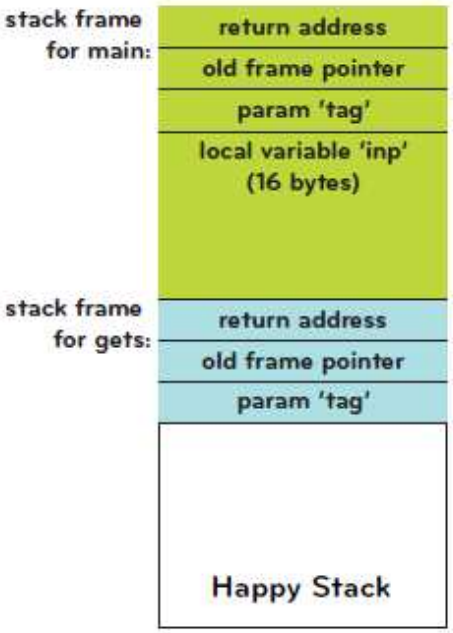
Recall: Programming languages store variables + state information on a stack.

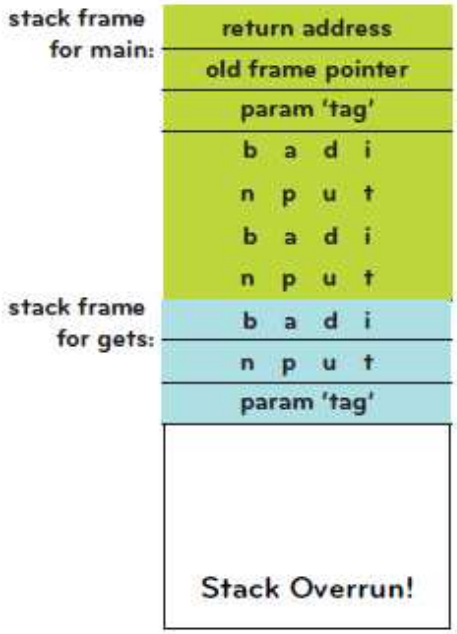
```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```



No bounds checking in gets():

```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

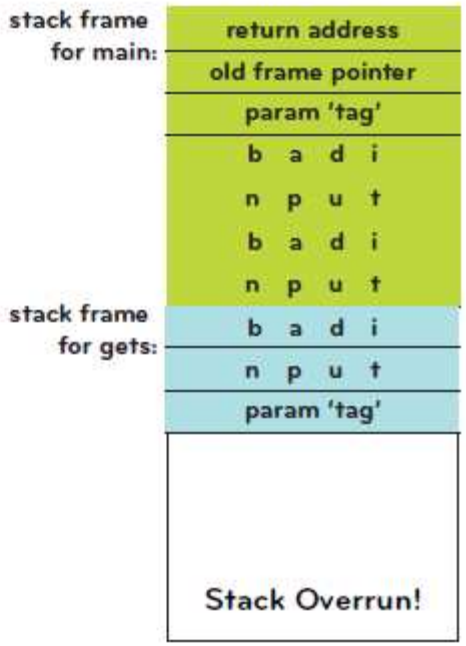




The result:

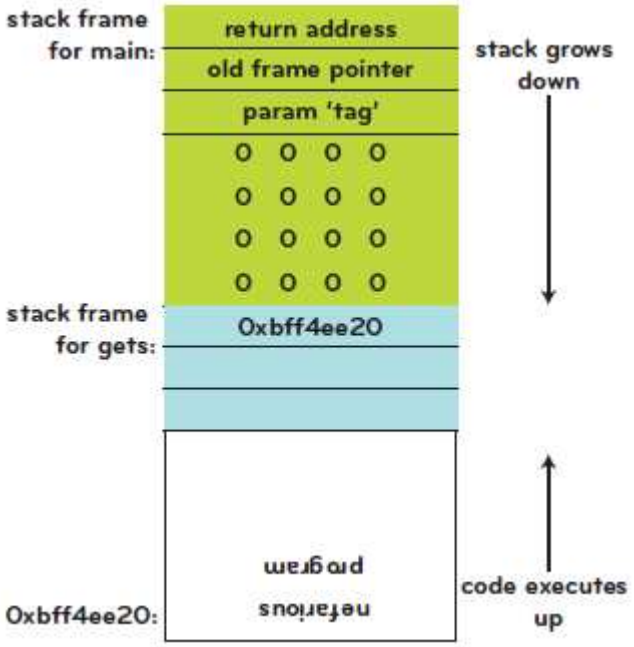
when gets() attempts to return to main, the program control goes to the address 0x62616469

Segmentation fault!



Smashing the stack

If we are really clever, can have the program execute something malicious in the stack.



Back to finger

If someone carefully crafted the input to finger, they could smash the stack and execute something else

→ say, 'sh'

(command prompt)

Example: The Morris Worm, 1988

Worm Propogation

Generically, a worm will:

- Find other systems to infect by looking at host tables
- Connect to other systems
- Copy itself over & invoke that copy

Morris Worm

Supposedly, this worm was intended to be a harmless tool to measure the internet at the time.

Instead, in just 3 days, it took out approximately 10% of all networked computers.

It used several known exploits, and contained a few bugs.

(99 lines of code!)

The worm tries 4 different attacks

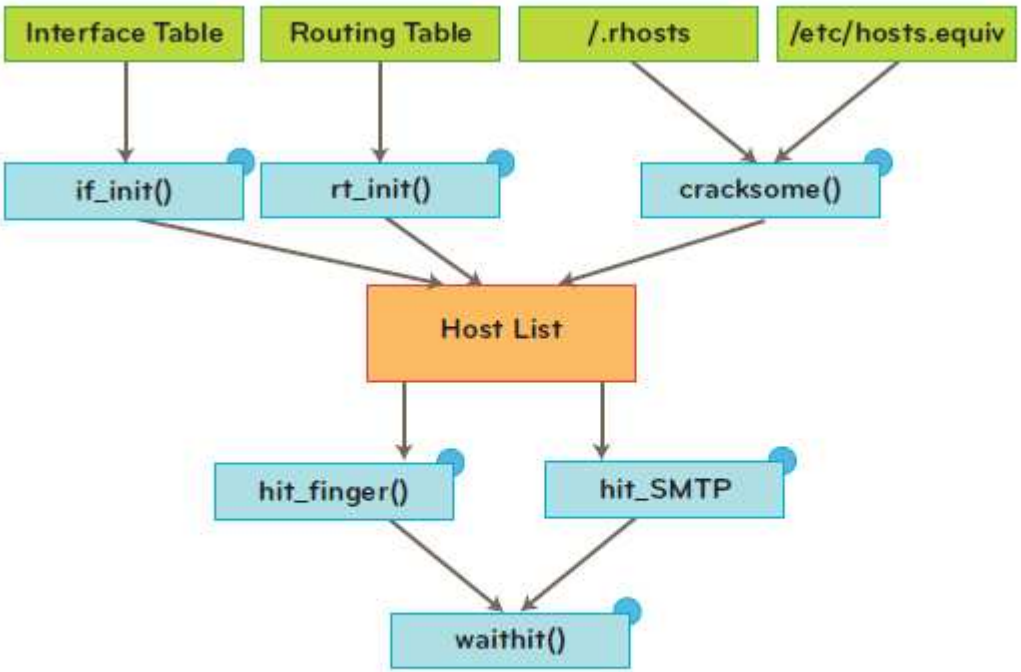
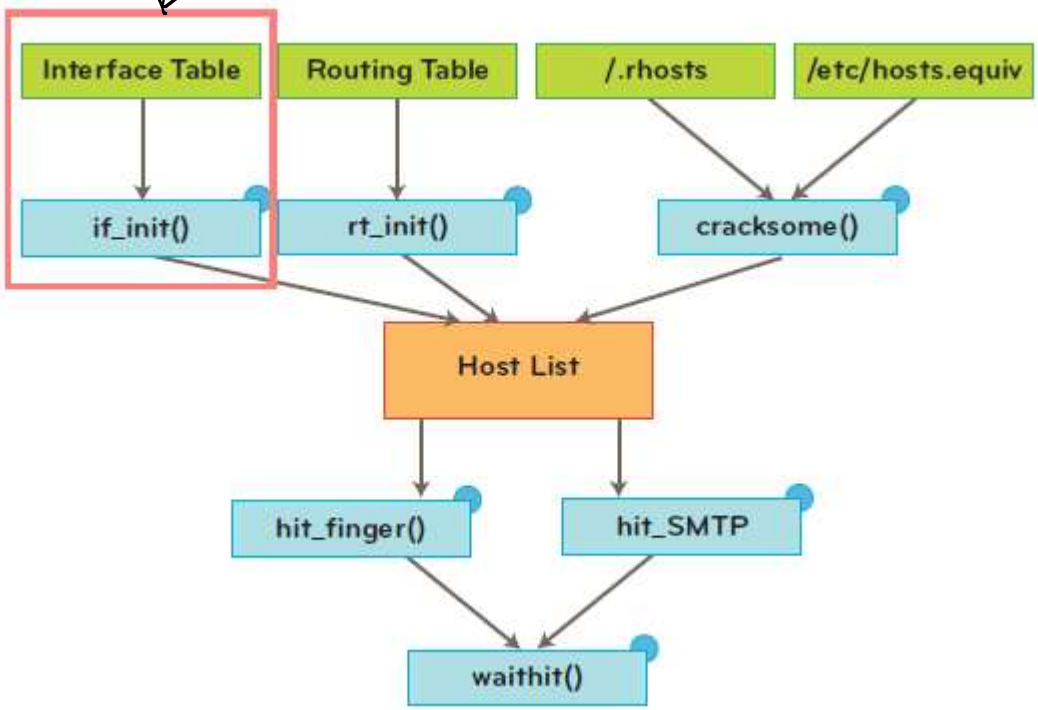


Figure adapted from "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988" by M. Eichin and J. Rochlis, 1989.

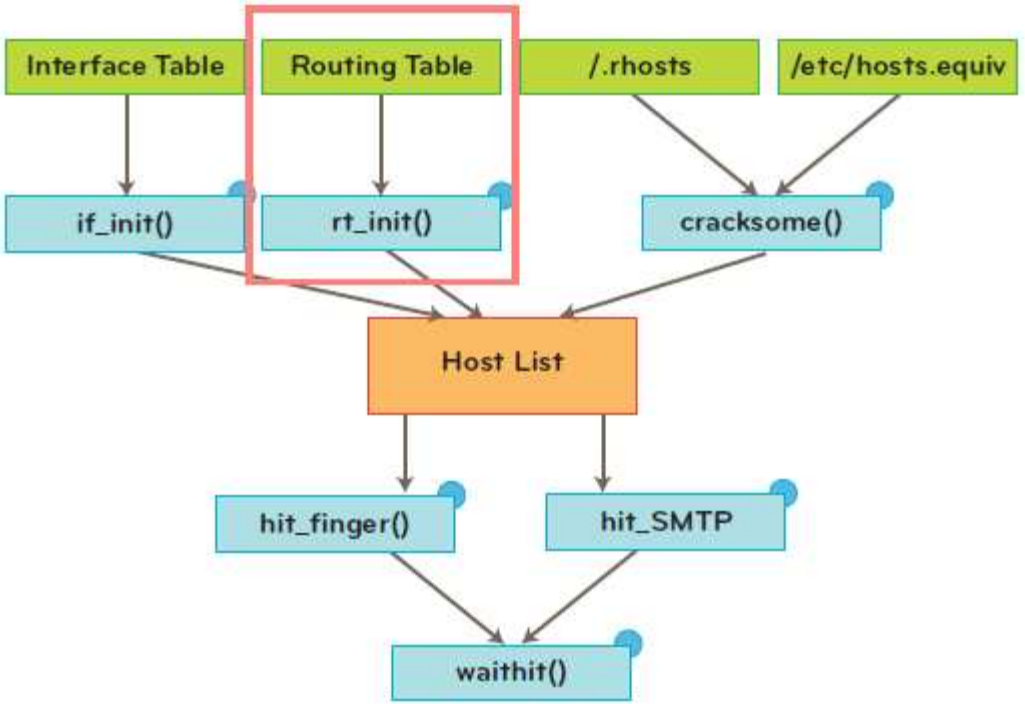
Step 1: Initialization: The worm "hides" itself

```
strcpy(argv[0], "sh");

struct rlimit rl;
rl.rlim_cur = 0;
rl.rlim_max = 0;
if (setrlimit(RLIMIT_CORE, &rl))
  ;
```

② Scan network interface



③ look up routing table

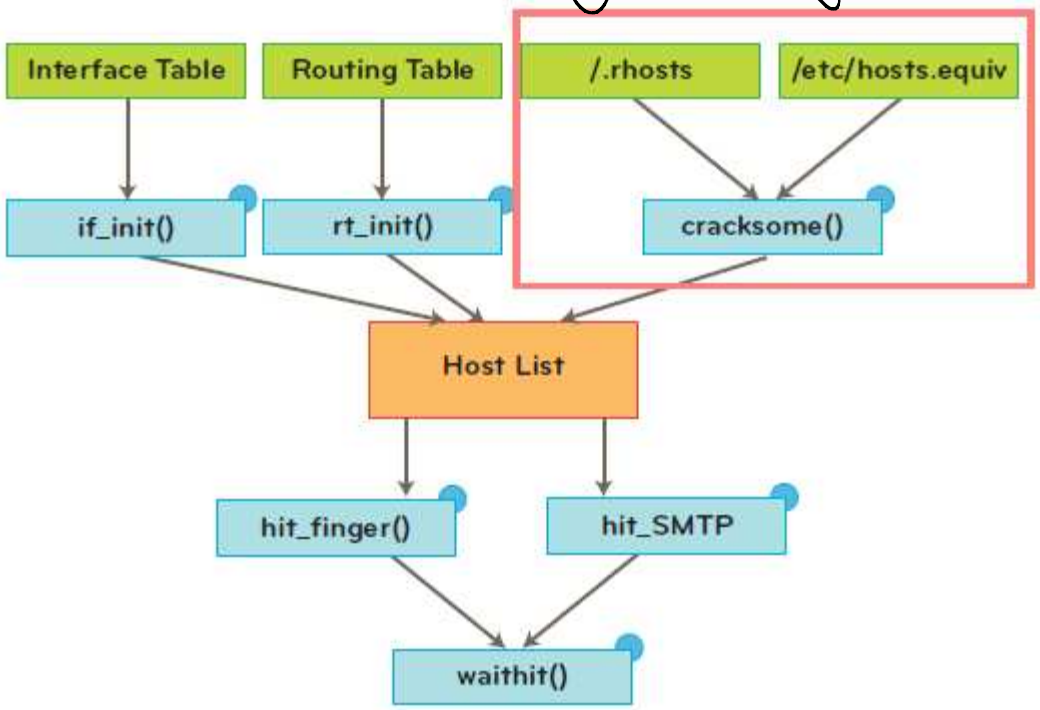


it does so using a call to netstat

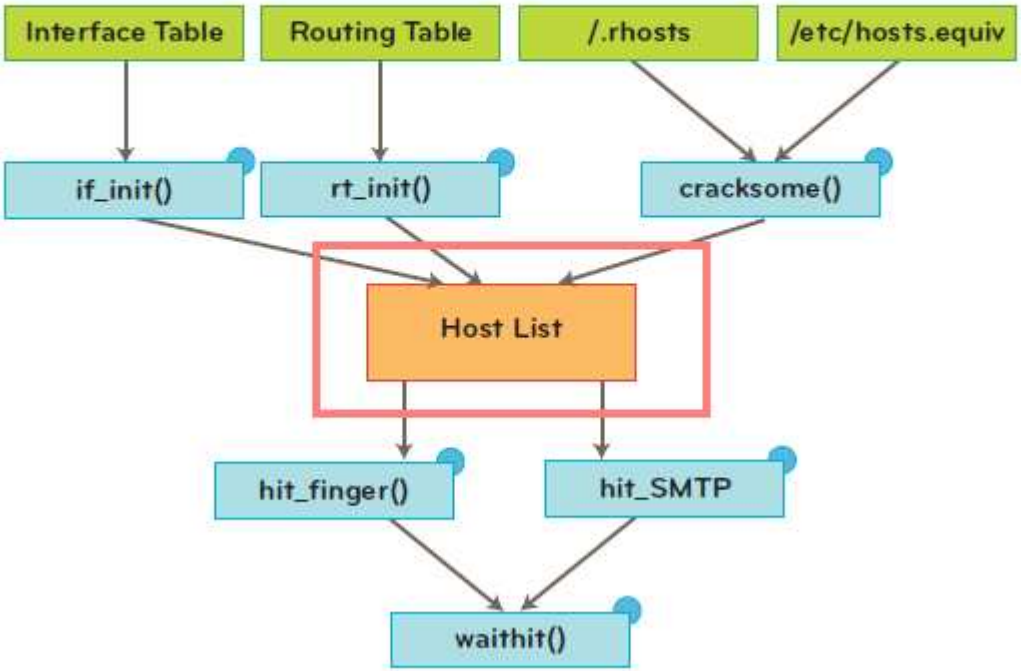
```
pipe = popen("/usr/ucb/netstat -r -n", "r");
```

```
Example output from my machine:
en220-m14560:~ crenshaw$ netstat -r -n
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif Expire
default            10.11.15.1         UGSc        9        4    en0
10.11.15/24        link#5             UCS        25        0    en0
10.11.15.1         0:0:c:7:ac:0       UHLW        1        0    en0   1199
10.11.15.14        0:26:4a:c:b1:4     UHLW        0        0    en0    611
10.11.15.15        0:21:86:ed:b:23    UHLW        0     5569    en0   1190
10.11.15.29        0:15:c5:2:56:ee    UHLW        0        0    en0   1058
10.11.15.31        0:25:bc:df:ff:86   UHLW        0        0    en0    714
10.11.15.42        0:21:86:fb:8c:82   UHLW        0        0    en0    858
10.11.15.45        0:21:86:ed:12:b3   UHLW        0        0    en0   1168
10.11.15.47        0:27:13:53:fa:b0   UHLW        0      104    en0    530
10.11.15.57        0:21:86:fb:89:5b   UHLW        0        0    en0   1090
10.11.15.67        0:13:20:49:8d:2e   UHLW        0        0    en0   1003
10.11.15.73        0:1f:5b:35:9d:10   UHLW        0        0    en0   1128
10.11.15.75        0:1f:5b:35:8f:58   UHLW        0        0    en0   1139
10.11.15.76        0:1f:5b:34:c5:58   UHLW        0        0    en0    860
10.11.15.81        0:1f:5b:35:67:d8   UHLW        0        1    en0   1197
10.11.15.87        127.0.0.1          UHS         0        9    lo0
10.11.15.100       0:26:4a:c:78:bc    UHLW        0        0    en0    616
10.11.15.108       5c:ff:35:4:98:9f   UHLW        0        0    en0   1079
10.11.15.122       0:21:86:fb:8c:a2   UHLW        0        0    en0   1074
10.11.15.255       link#5             UHLWb       3      295    en0
```

④ Find likely targets



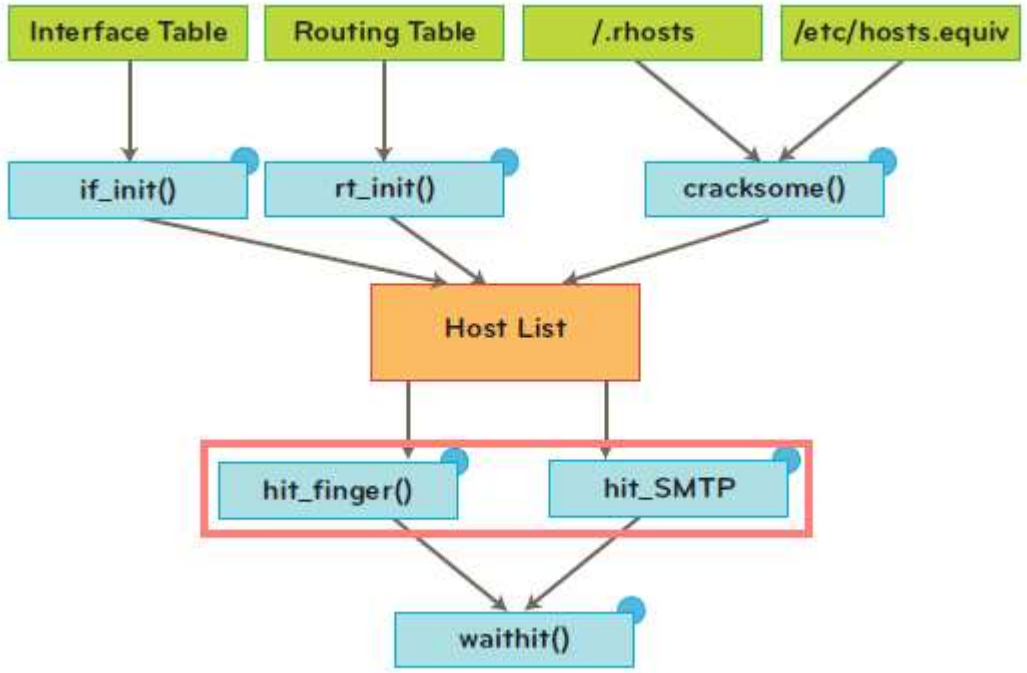
at this point, a list of potential targets has been created.



Attack!

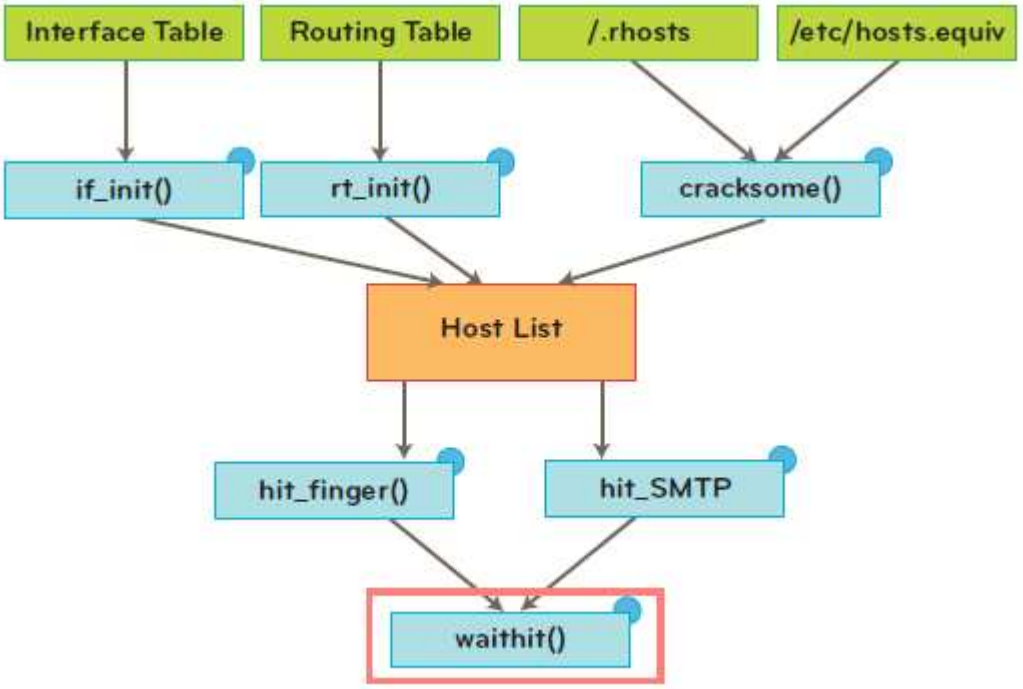
Try finger + Sendmail exploit on the list of possible hosts.

A piece of code (that is very portable) called l1.c 'ls' then copied to the other host.



A program called waithit() then transfers the rest of the code over.

And it all begins again...



The other 2 attack exploited common passwords, and vulnerabilities in rsh/exec.

All of these attacked only DEC VAX machines running BSD or SUN.

didn't work

Morris Worm (Recap)

Remember, this was only supposed to measure the internet.

What went wrong?

The code contained several bugs which turned it into the first large scale attack.

① The worm was supposed to detect if it was already on a machine.

This broke, so machines were reinfected.

This crashed any infected machine.

② The worm was supposed to send a packet of data back to 128.32.137.13.

This broke too.

The Result

"Robert T. Morris was convicted of violating the Computer Fraud and Abuse Act (Title 18), and sentenced to three years of probation, 400 hours of community service, a fine of $10,050, and the costs of his supervision."

He's now an assistant professor at MIT, conducting research in computer networks and operating systems.

Robert T. Morris, Today
Photograph from MIT website

(Amusingly, he was a Cornell grad student, but somehow originally released the worm at MIT.)

Why study this?

1988 was a long time ago.

If we no longer study the Apple IIe, why study this?

2 reasons I can think of:

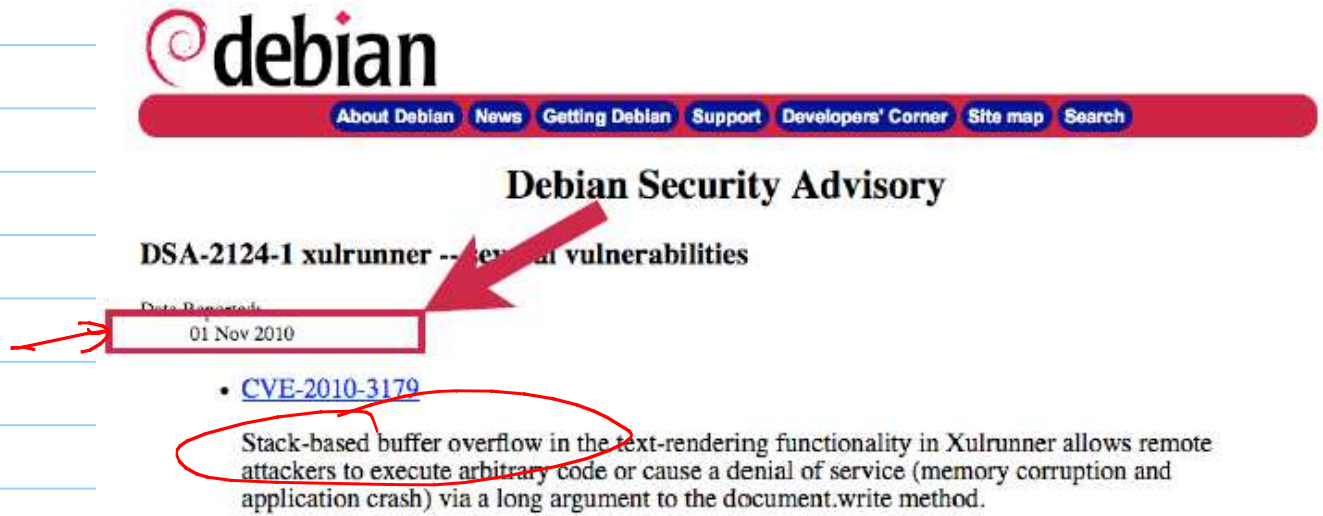
Historical Relevance

The Morris Worm was the reason the US began setting up legislation to control computer crime.

This worm was the motivation behind founding CERT, the organization responsible for monitoring & tracking network emergencies.

Stack-Overflow Techniques

Still very relevant!



debian

About Debian | News | Getting Debian | Support | Developers' Corner | Site map | Search

Debian Security Advisory

DSA-2124-1 xulrunner -- [illegible] vulnerabilities

Date Reported:
01 Nov 2010

- CVE-2010-3179

Stack-based buffer overflow in the text-rendering functionality in Xulrunner allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption and application crash) via a long argument to the document.write method.

Next time: (& future topics)

- Network defenses
- Constructing attacks (DETER)
- Secure coding & auditing
- New versus old techniques (& why things changed)
- Web applications
- Good practices (as admin or individual)