Operating Systems Design
20. Security

Paul Krzyzanowski
pxk@cs.rutgers.edu

Protection & Security

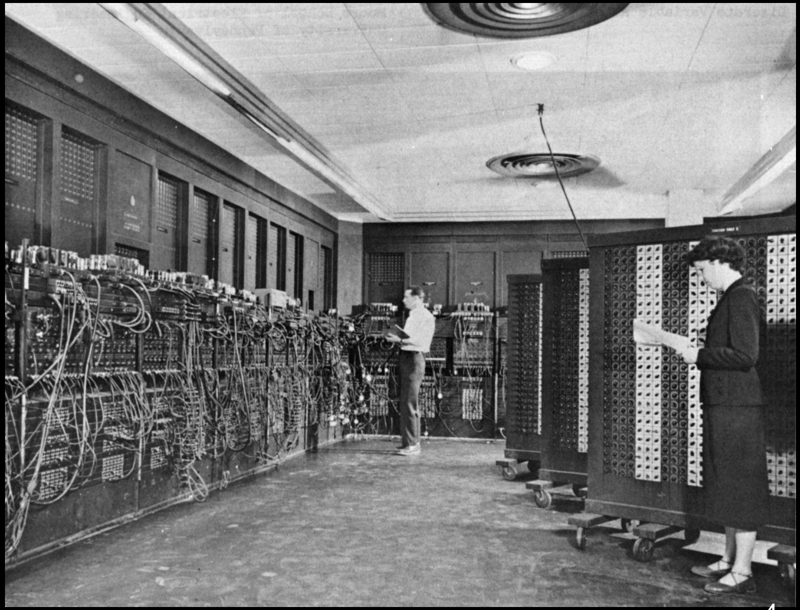
- Security
  - Prevention of unauthorized access to a system
    - Malicious or accidental access
    - “access” may be:
      - user login, a process accessing things it shouldn’t, physical access
    - The access operations may be reading, destruction, or alteration
- Protection
  - The mechanism that provides and enforces controlled access of resources to processes
  - A protection mechanism *enforces* security policies

Threats

Computer security… then

Issue from the dawn of computing:

- Colossus at Bletchley Park: breaking codes
- ENIAC at Moore School: ballistic firing tables
- single-user, single-process systems
- data security needed
- physical security



Computer security… now

- Sensitive data of different users lives on the same file servers
- Multiple processes on same machine
- Authentication and transactions over network
  - open for snooping
- We might want to run other people’s code in our process space
  - Device drivers, media managers
  - Java applets, games
  - not just from trusted organizations

Systems are easier to attack

Automation

- Data gathering
- Mass mailings

Distance

- Attack from your own home

Sharing techniques

- Virus kits
- Hacking tools

Penetration

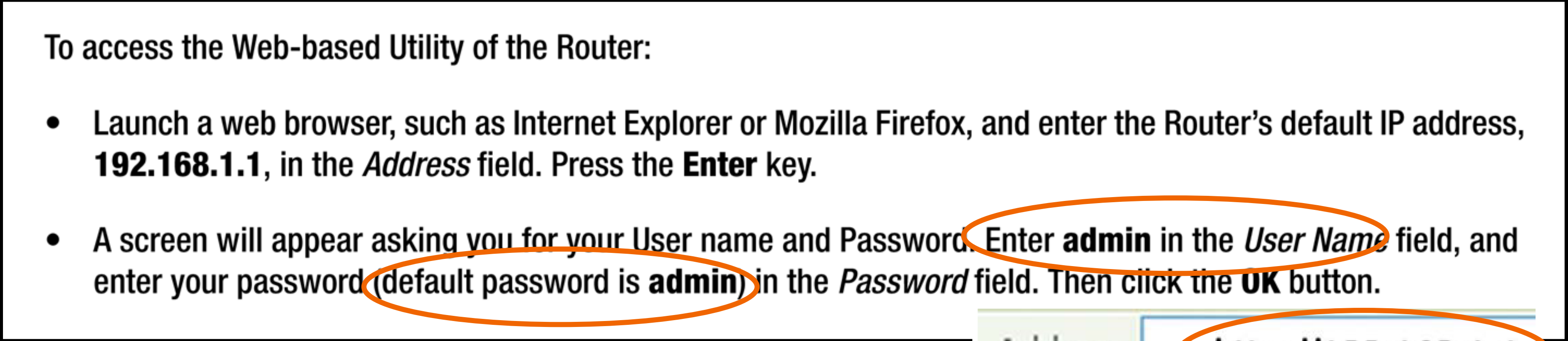
Guess a password

- system defaults, brute force, dictionary attack

Crack a password

- Online vs offline
- Precomputed hashes (see rainbow tables)
  - Defense: Salt

Penetration: Guess/get a password



To access the Web-based Utility of the Router:

- Launch a web browser, such as Internet Explorer or Mozilla Firefox, and enter the Router's default IP address, **192.168.1.1**, in the *Address* field. Press the **Enter** key.
- A screen will appear asking you for your User name and Password. Enter **admin** in the *User Name* field, and enter your password (default password is **admin**) in the *Password* field. Then click the **OK** button.

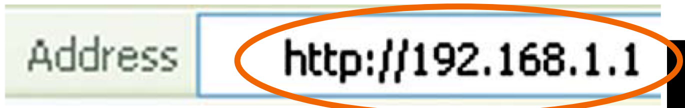
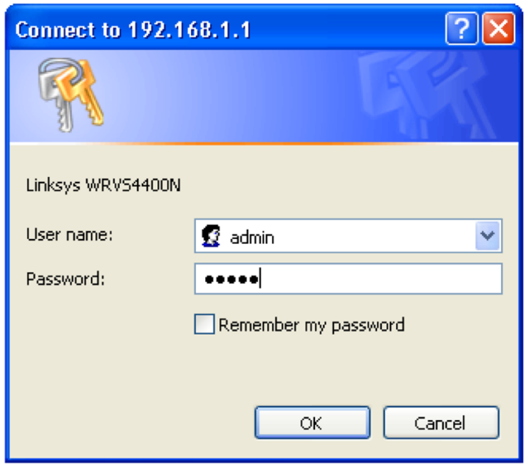


Figure 6-1: Router's IP Address



Linksys WRVS4400N

User name: admin

Password: •••••

☐ Remember my password

Figure 6-2: Login Screen for Web-based Utility

Page 29 of the *Linksys Wireless-N Gigabit Security Router with VPN* user guide

Penetration

Social engineering

- people have a tendency to trust others
- *finger* sites – deduce organizational structure
- facebook, twitter, blogs, personal home pages
- look through dumpsters for information
- impersonate a user
- Phishing: impersonate a company/service

Penetration

Trojan horse

- program masquerades as another
- Get the user to click on something, run something, enter data

```
*****************************************************************

  The DCS undergrad machines are for DCS coursework only.

*****************************************************************

  Getting "No valid accounts?" Go to
  http://remus.rutgers.edu/newaccount.html
  and add yourself back.

login: pxk
Password:
Login incorrect
```

Trojan horse

Disguising error messages

New Windows XP SP2 vulnerability exposed

ZDNet UK

Munir Kotadias
ZDNet Australia
November 22, 2004, 12:50 GMT

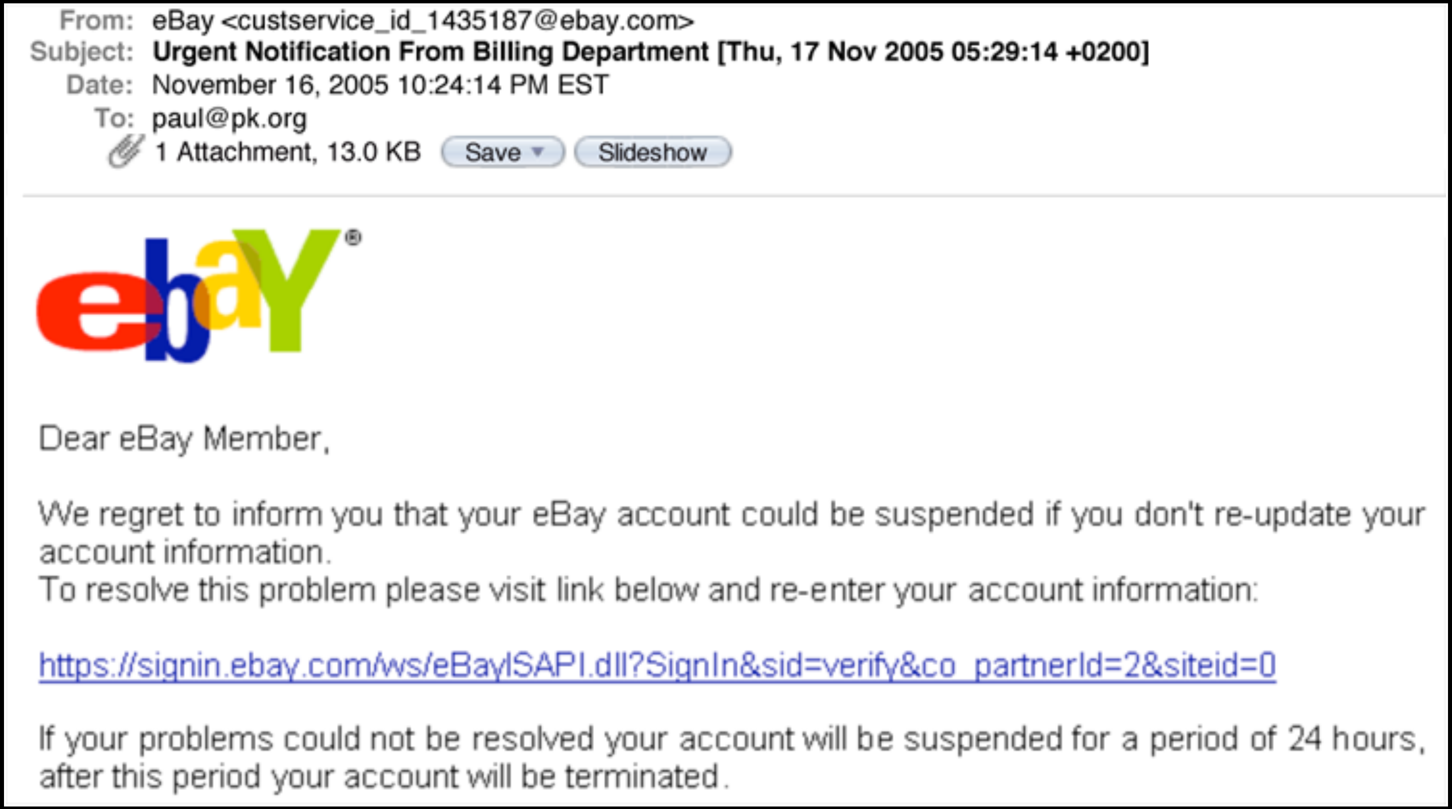
A vulnerability in Microsoft's Windows XP SP2 can allow an executable file to be run by hackers on target machines, according to security researchers

… it is possible to craft a special error message that is able to bypass a security function in IE that was created to warn users before they download potentially harmful content. … a malicious Web site could prompt all its visitors with a standard grey dialogue box welcoming a user to the site before allowing access to the site's content. If a user clicks on the welcome box they could unknowingly install a file that gives control of their computer to a third party.

`http://tinyurl.com/5mj9f`

Phishing

Masqueraded e-mail



From: eBay <custservice_id_1435187@ebay.com>
Subject: **Urgent Notification From Billing Department [Thu, 17 Nov 2005 05:29:14 +0200]**
Date: November 16, 2005 10:24:14 PM EST
To: paul@pk.org
1 Attachment, 13.0 KB

Dear eBay Member,

We regret to inform you that your eBay account could be suspended if you don't re-update your account information.
To resolve this problem please visit link below and re-enter your account information:

https://signin.ebay.com/ws/eBayISAPI.dll?SignIn&sid=verify&co_partnerId=2&siteid=0

If your problems could not be resolved your account will be suspended for a period of 24 hours, after this period your account will be terminated.

Malicious Files and Attachments

Take advantage of:

- Programs that automatically open attachments
- Systems that hide extensions yet use them to execute a program – trick the user

`love-letter.txt.vbs` *looks like* `love-letter.txt`

`resume.doc.scr` *looks like* `resume.doc`

Exploiting bugs

Exploit software bugs

- Most (all) software is buggy
- Big programs have lots of bugs
  - *sendmail*, *wu-ftp*
- some big programs are *setuid* programs
  - *lpr, uucp, sendmail, mount, mkdir, eject*

Common bugs

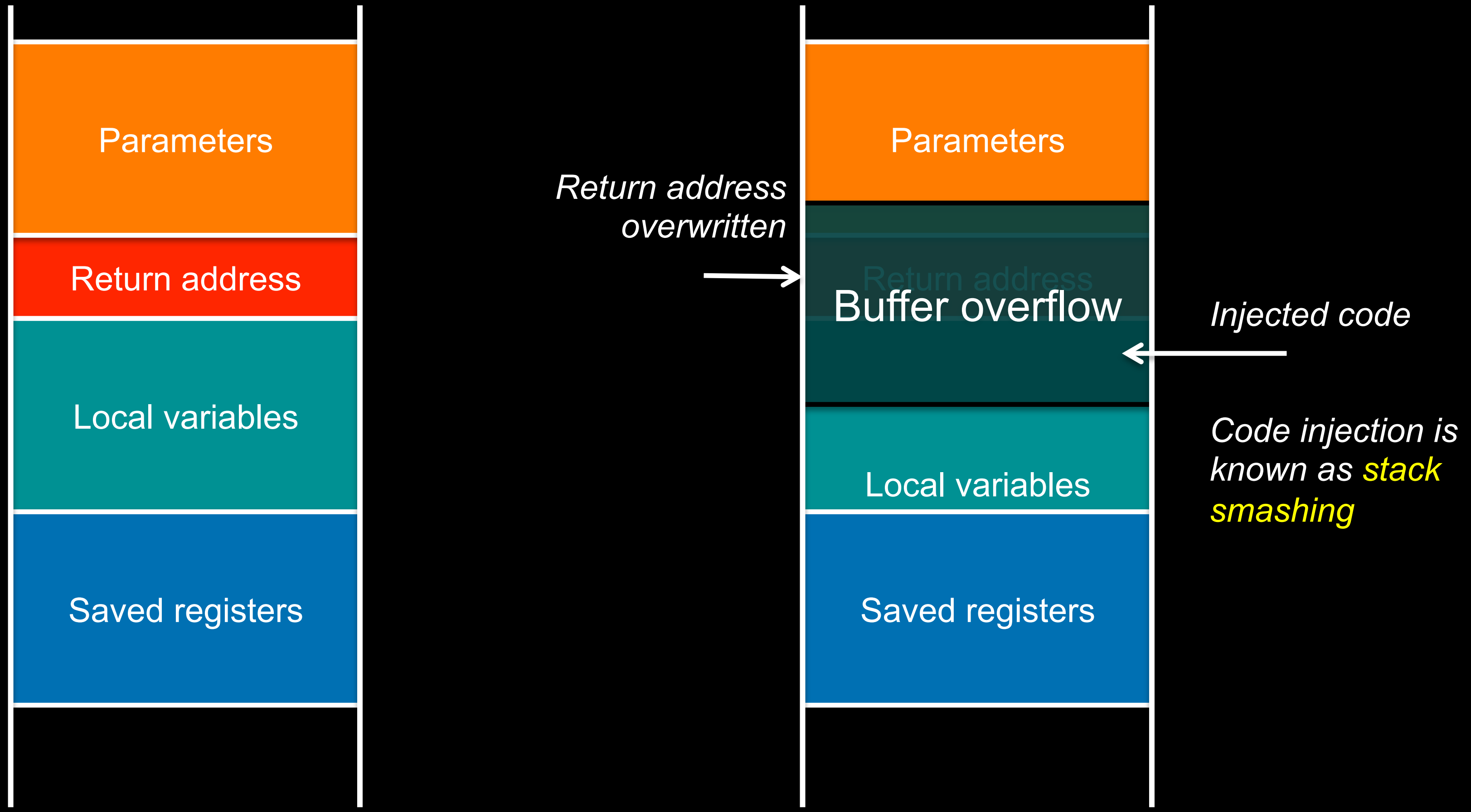
- buffer overflow
  (blindly read data into buffer)
  - e.g., *gets*
- back doors and undocumented options

The classic buffer overflow bug

gets.c from V6 Unix:

```
gets(s)
char *s;
{ /* gets (s) - read a string with cgetc and store in s */
   char *p;
   extern int cin;
   if (nargs () == 2)
    IEHzap("gets  ");
   p=s;
   while ((*s = cgetc(cin)) != '\n' && *s != '\0')
    s++;
   if (*p == '\0') return (0);
   *s = '\0';
   return (p);
}
```

Buffer overflow



More data was input than the programmer expected, causing the local array that was allocated for the data to overflow. The overflow overwrites the return address on the stack. Now, when the function returns, the return address is under the control of the attacker.

Dealing with buffer overflows: No Execute

- Executable space protection
  - Disallow code execution on the stack or heap
  - Set MMU per-page execute permissions to no-execute
  - Iintel and AMD added this support in 2004

  - Examples
    - Microsoft DEP (Data Execution Prevention) (since XP SP2)
    - Linux PaX patches
    - OS X ≥10.5

Return Oriented Programming (ROP)

- Stack can still be corrupted
- Can overwrite return address with address of a library function
  - Does not have to be the start of the library routine
    - "borrowed chunks"
  - When the library hits the RET instruction, that return address is on the stack, under the attacker's control
- ROP chains together sequences ending in RET
  - Build together "gadgets" for arbitrary computation
  - Buffer overflow contains a sequence of addresses that direct each successive RET instruction
- Make attacking easier: C compiler that generates gadgets!

Dealing with buffer overflows: ASLR

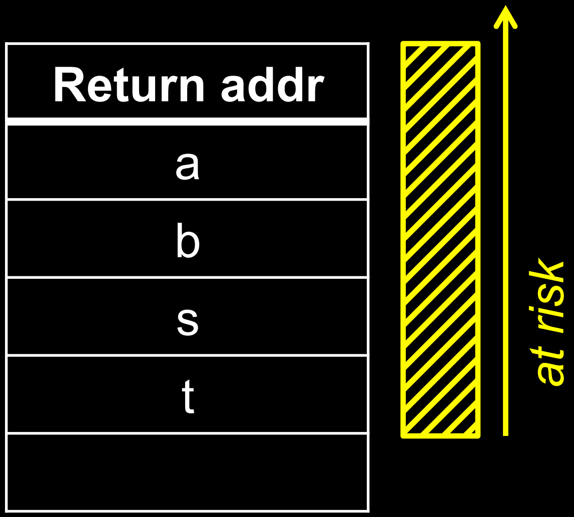
- Address Space Layout Randomization
  - Dynamically-loaded libraries used to be loaded in the same place each time … ditto for the stack & memory-mapped files
  - Well-known locations make them branch targets in a buffer overflow attack
  - Position the stack and memory-mapped files (including libraries) to random locations
  - Implemented in
    - OpenBSD, Windows Vista+, Windows Server 2008, Linux 2.6.15, OS X

Dealing with buffer overflows: Canaries

- Stack canaries
  - Place a random integer before the return address on the stack
  - Before a return, check that the integer is there and not overwritten: a buffer overflow attack will likely overwrite it

```
int a, b=999;
char s[5], t[7];

gets(s);
```



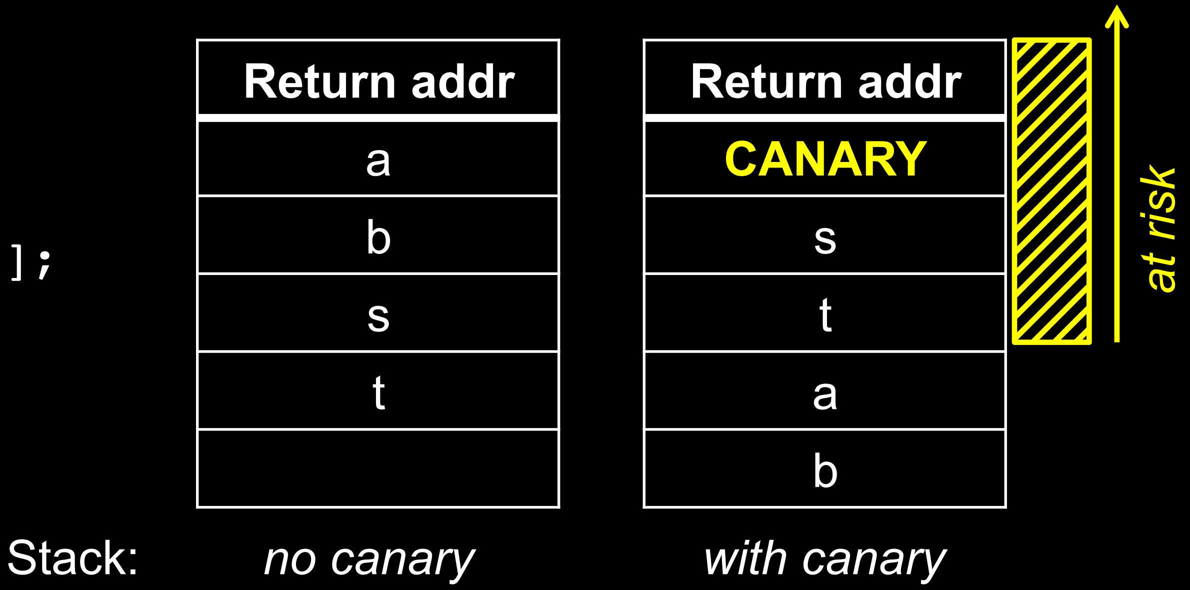
Stack: *no canary*

Dealing with buffer overflows: Canaries

- Stack canaries
  - Place a random integer before the return address on the stack
  - Before a return, check that the integer is there and not overwritten: a buffer overflow attack will likely overwrite it
  - Allocate arrays into higher memory in the stack so they won't clobber other automatic variables

```
int a, b=999;
char s[5], t[7];

gets(s);
```



Threats (continued)

Virus

- Does not run as a self-contained process
- Code is attached onto another program or script

- File infector
  - primarily a problem on systems without adequate protection mechanisms
- Boot-sector
- Macro (most common … visual basic scripting)
- Hypervisor
  - install on virtual machines

Virus scanning

- Search for a “signature”
  - Stream of bits in a virus that (we hope!) is unique to the virus and not any legitimate code
  - *NOT a cryptographic signature!*
- Some viruses are encrypted
  - Signature is either the code that does the decryption or the scanner must be smart enough to decrypt the virus
- Some viruses mutate to change their code every time they infect another system
  - Run the code through an emulator to detect the mutation

Virus scanning

- You don’t want to scan through hundreds of thousands of files
  - Search in critical places likely to be infected (e.g., \windows\system32 or removable media)
- Passive disk scan vs. active I/O scan … or both
- “Zero-Day Threats”: new virus – signature unknown
- Virus scanning is becoming less effective
  - Estimates are that only 10-30% of new viruses in the network are detectable

Key loggers

- Record every keystroke
- Windows *hook* mechanism
  - Procedure to intercept message traffic before it reaches a target windows procedure
  - Can be chained
  - Installed via *SetWindowsHookEx*
  - WH_KEYBOARD and WH_MOUSE
    - Capture key up, down events and mouse events
- Hardware loggers

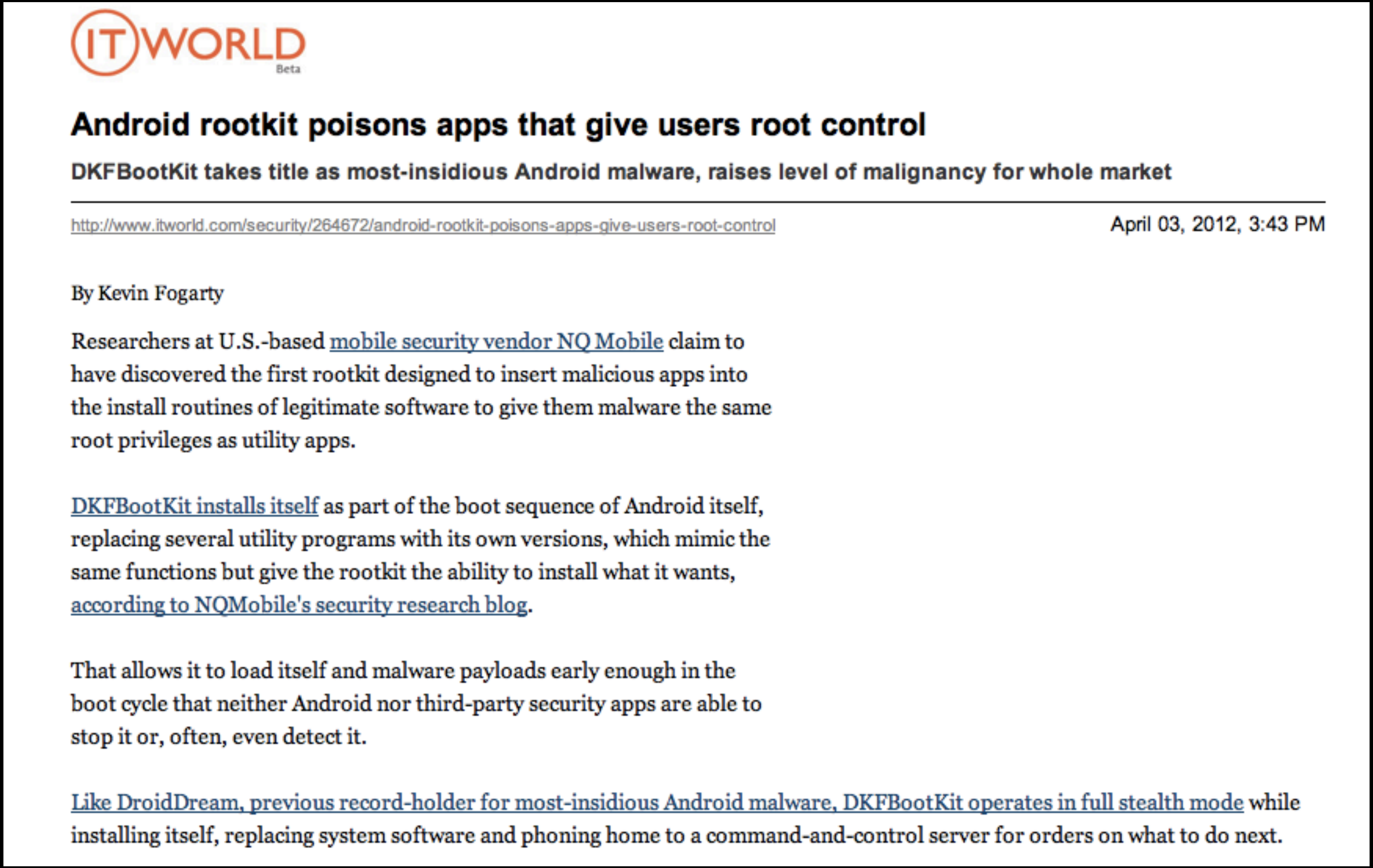
Rootkits

- Replacement commands (or standard shared libraries or OS components) to hide the presence of an intruder
  - ps, ls, who, netstat, …
- Hide the presence of a user or additional software (backdoors, key loggers, sniffers)
- Now the OS can no longer be trusted!

E.g., Sony BMG DRM rootkit (October 2005)

- Creates hidden directory; installs several of its own device drivers; reroutes Windows system calls to its own routines
- Intercepts kernel-level APIs and disguises its presence with cloaking (hides sys files)

Rootkits



IT WORLD Beta

Android rootkit poisons apps that give users root control

DKFBootKit takes title as most-insidious Android malware, raises level of malignancy for whole market

http://www.itworld.com/security/264672/android-rootkit-poisons-apps-give-users-root-control

April 03, 2012, 3:43 PM

By Kevin Fogarty

Researchers at U.S.-based mobile security vendor NQ Mobile claim to have discovered the first rootkit designed to insert malicious apps into the install routines of legitimate software to give them malware the same root privileges as utility apps.

DKFBootKit installs itself as part of the boot sequence of Android itself, replacing several utility programs with its own versions, which mimic the same functions but give the rootkit the ability to install what it wants, according to NQMobile's security research blog.

That allows it to load itself and malware payloads early enough in the boot cycle that neither Android nor third-party security apps are able to stop it or, often, even detect it.

Like DroidDream, previous record-holder for most-insidious Android malware, DKFBootKit operates in full stealth mode while installing itself, replacing system software and phoning home to a command-and-control server for orders on what to do next.

Dealing With Rootkits

- Hope you don’t get one!
- Restrict permission to modify system files
  - To avoid installing a rootkit in the first place
  - But users often grant permissions during installation
    - And permissions may be needed for drivers
- Signed software and operating system components
  - Microsoft Vista & Windows 7:
    - Requires kernel-mode software to have a digital signature (x64-based systems only)
- Tripwire
  - Software to monitor for changes in files and components in a system

Sandboxing

Restricting what an app can do

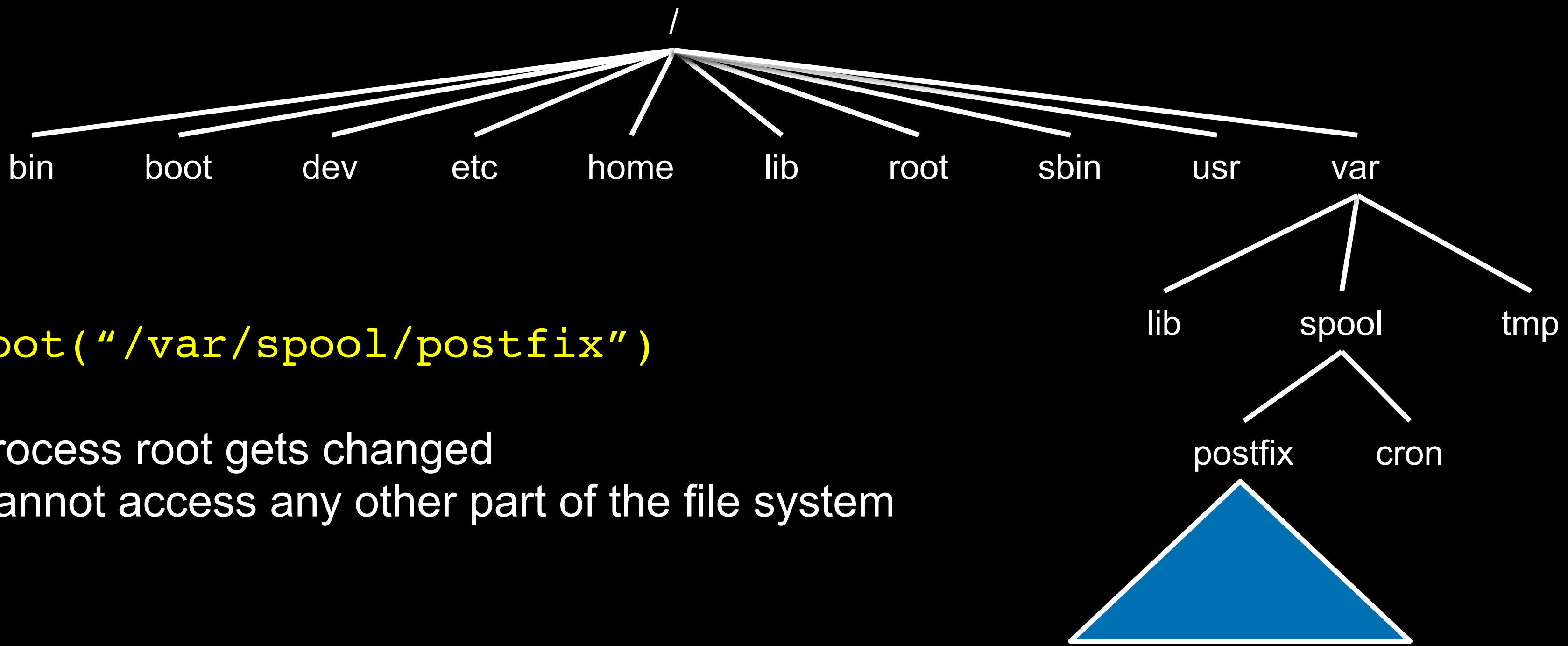
- Traditional OS approach to protection
  - Privileges are an attribute of the user
  - Every process the user runs has the same privileges
    - A program can delete all of your files
      … or upload them to a remote server

- *How do we protect ourselves from malicious applications?*

- Mandatory Access Control
  - Restricts information flow between classes of users
  - Doesn't solve the problem

A basic approach: chroot jail

Chroot() system call: change the root directory for a process



```
chroot("/var/spool/postfix")
```

- Process root gets changed
- Cannot access any other part of the file system

Sandboxing

- Per process access control: restrict what applications can do
- Examples
  - *An app shouldn't read /etc/passwd*
  - *An app shouldn't write any files*
  - *An app should not establish a TCP/IP connection*
  - *An app shouldn't access your contacts*

- Sandboxing is used in:
  - Google Chrome browser
  - Microsoft Office 2010 Protected View
  - Apple iOS sandboxing
  - Apple XNU Sandbox framework
  - SELinux
  - FreeBSD TrustedBSD system (mostly MAC)
  - Windows: access control at kernel object level with inherited permissions

Example: Apple Sandbox

- Create a list of rules that is consulted to see if an operation is permitted
- Components:
  - Set of libraries for initializing/configuring policies per process
  - Server for kernel logging
  - Kernel extension using the TrustedBSD API for enforcing individual policies
  - Kernel support extension providing regular expression matching for policy enforcement
- *sandbox-exec* command & sandbox_init function
  - sandbox-exec: alls sandbox_init before fork() and exec()
  - `sandbox_init(kSBXProfileNoWrite, SANDBOX_NAMED, errbuf);`

Apple sandbox setup & operation

- *sandbox_init*:
  - Convert human-readable policies into a binary format for the kernel
  - Policies passed to the kernel to the TrustedBSD subsystem
  - TrustedBSD subsystem passes rules to the kernel extension
  - Kernel extension installs sandbox profile rules for the current process

- Operation
  - System calls hooked by the TrustedBSD layer will pass through Sandbox.kext for policy enforcement
  - The extension will consult the list of rules for the current process
  - Some rules require pattern matching (e.g., filename pattern)

Apple sandbox policies

- Some pre-written profiles:
  - Prohibit TCP/IP networking
  - Prohibit all networking
  - Prohibit file system writes
  - Restrict writes to specific locations (e.g., /var/tmp)
  - Perform only computation: minimal OS services

Java Sandbox

Java Virtual Machine

1. Bytecode verifier: verifies Java bytecode before it is run
   - Disallow pointer arithmetic
   - Automatic garbage collection
   - Array bounds checking
   - Null reference checking
2. Class loader: determines if an object is allowed to add classes
   - Ensures key parts of the runtime environment are not overwritten
   - Runtime data areas (stacks, bytecodes, heap) are randomly laid out
3. Security manager: enforces *protection domain*
   - Defines the boundaries of the sandbox (file, net, native, etc. access)
   - Consulted before any access to a resource is allowed

All bets are off if you allow native methods!

The End