

# Operating Systems Design

## 20. Security

Paul Krzyzanowski  
pxk@cs.rutgers.edu

# Threats

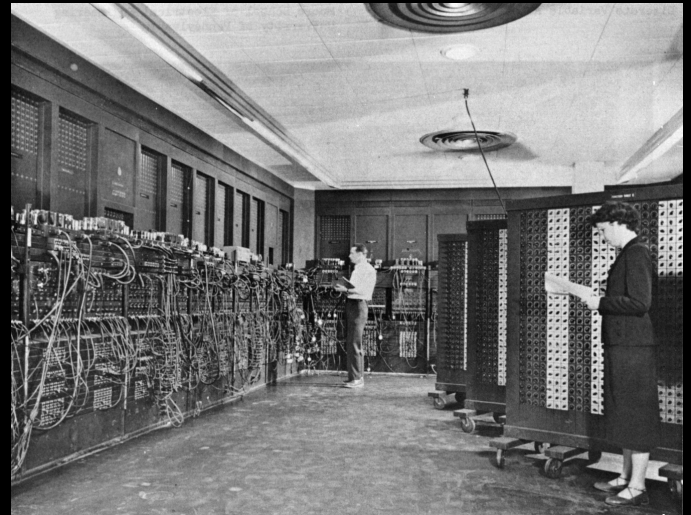
Adversary

# Computer security... then

---

Issue from the dawn of computing:

- 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

White hat

Black hat  
hacking

# Penetration



## Guess a password

- system defaults, brute force, dictionary attack

## Crack a password

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

123456  
password  
Letmein ←

Balloon

Apple  
Apple123  
123456  
Balloon

A  
AA  
→ AAA  
:  
→ A2ADD  
BBA  
Balloon ✓

Jack the ripper

Apple	[redacted]
Apple123	ACD27DE ...
123456	:
Balloon	:
	:

# 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

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

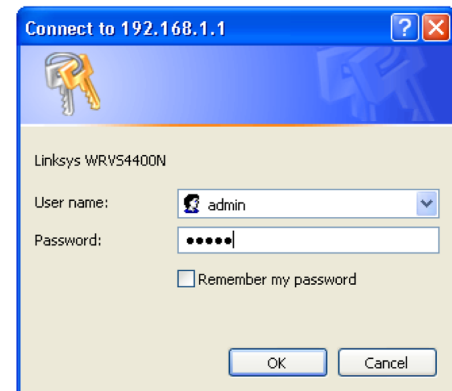


Figure 6-2: Login Screen for Web-based  
Utility

# Penetration

---

People  
**Social engineering** → Situation

- people have a tendency to trust others
- 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

*malicious*

*benign*



\*\*\*\*\*

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:

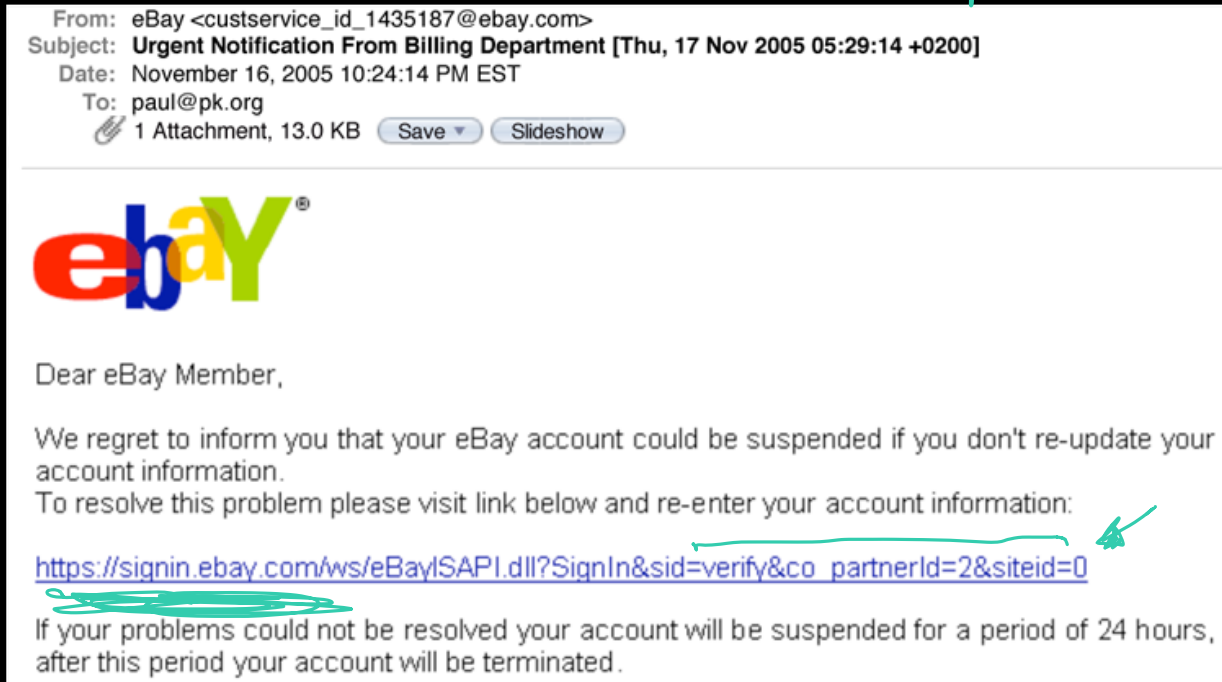
Login incorrect



# Phishing

## Masqueraded e-mail

one password 1 password  
last pass



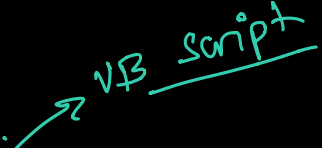
# Malicious Files and Attachments

---

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



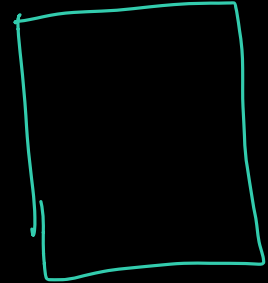
resume.doc.scr *looks like* resume.doc

# Exploiting bugs

---

## Exploit software bugs

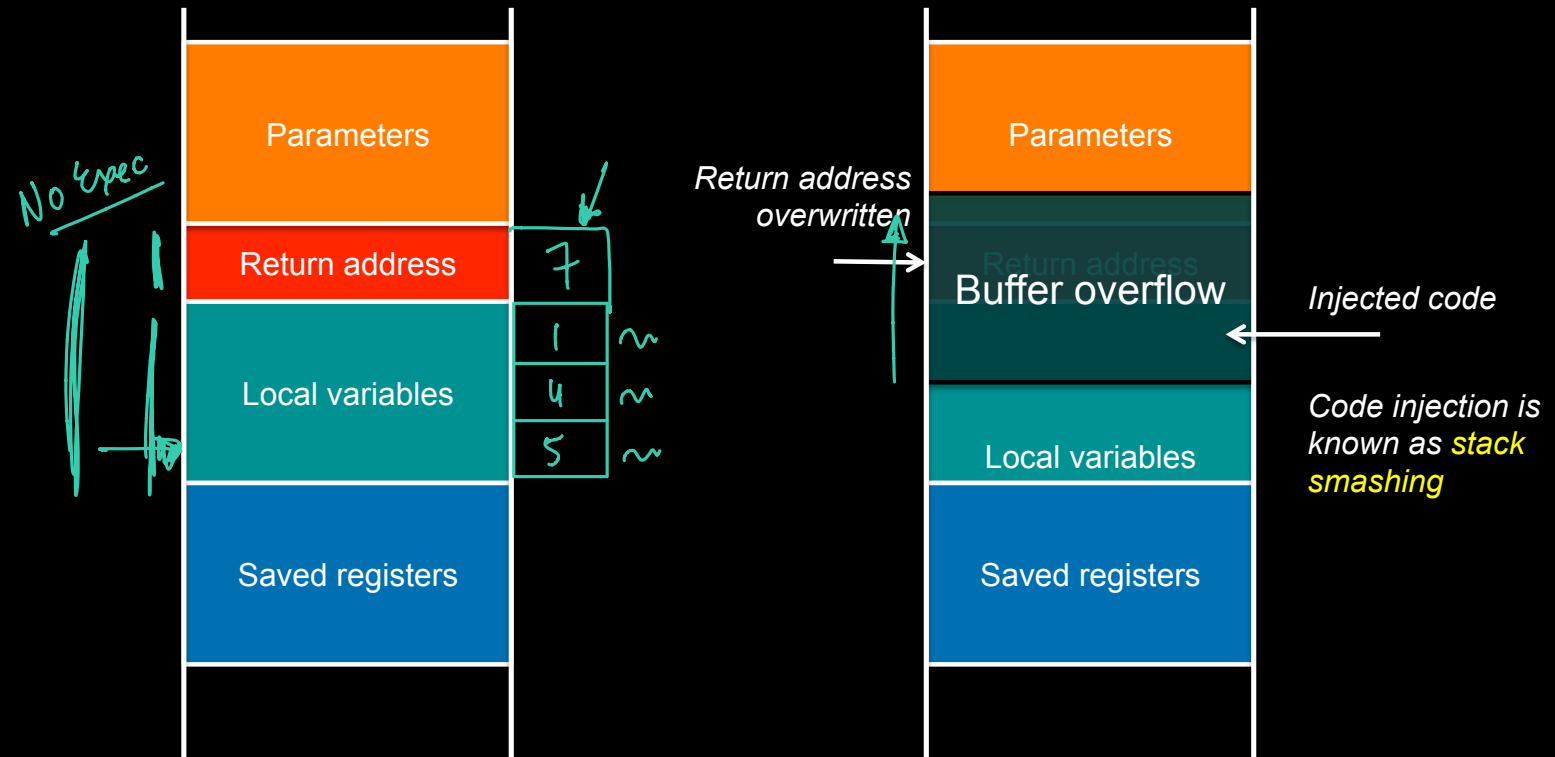
- Most (all) software is buggy
- Big programs have lots of bugs
  - sendmail, wu-ftp
- some big programs are ~~setuid programs~~
  - lpr, uucp, sendmail, mount, mkdir, eject



## Common bugs

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

# Buffer overflow



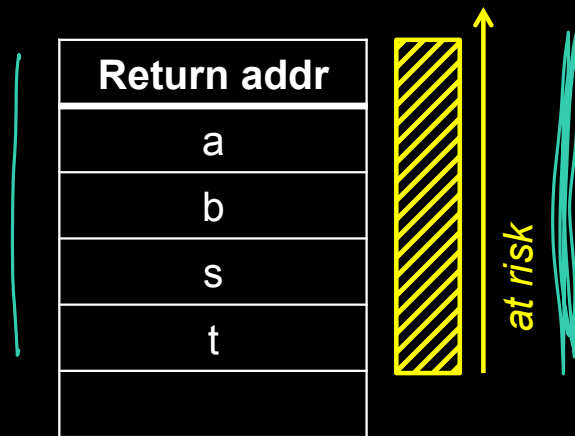
More data was input than the programmer expected, causing the local array that was allocated for the data to overflow. The overflow overwrites the return address on the stack. Now, when the function returns, the return address is under the control of the attacker.

# Dealing with buffer overflows: Canaries

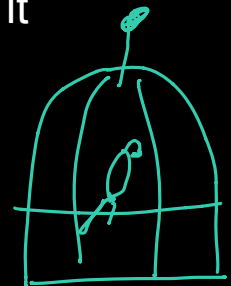
- Stack canaries

- 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

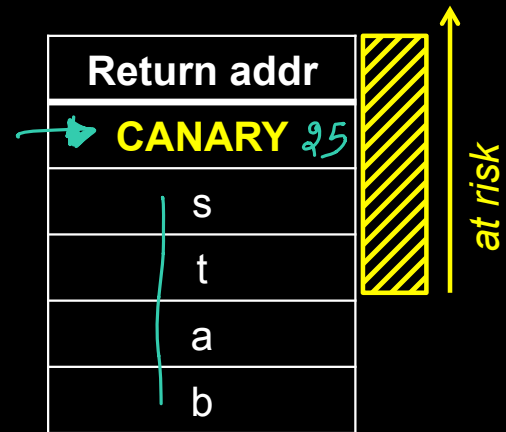
```
int a, b=999;  
char s[5], t[7];  
  
gets(s);
```

Stack:

*no canary*

Return addr
a
b
s
t

*with canary*

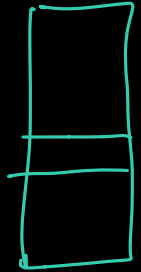


# Threats (continued)

# 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

10011001100

- Search for a “signature”
  - Stream of bits in a virus that (we hope!) is unique to the virus and not any legitimate code

Behavior

- 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

# Key loggers



---

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



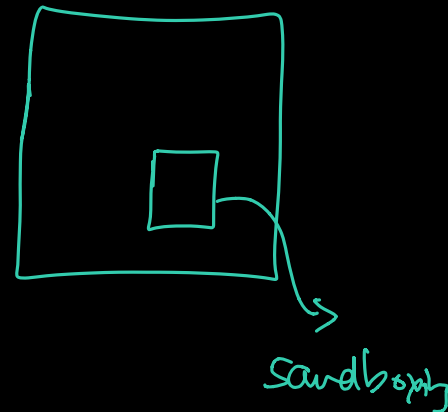
# Rootkits

---

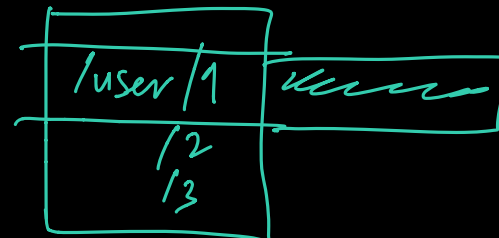
- Replacement commands (or standard shared libraries or OS components) to hide the presence of an intruder
  - ps, ls, who, netstat, ...  

- Hide the presence of a user or additional software (backdoors, key loggers, sniffers)
- Now the OS can no longer be trusted!  


# Dealing With Rootkits

- Hope you don't get one!
- Restrict permission to modify system files



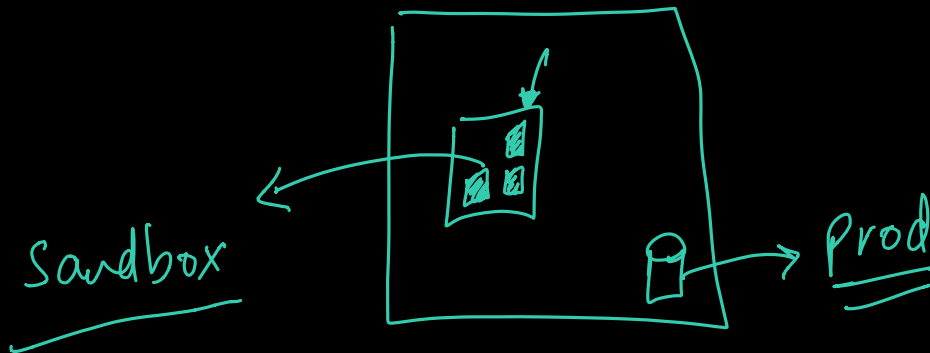
- Signed software and operating system components



- Tripwire
  - Software to monitor for changes in files and components in a system



# Sandboxing



# The End