Security

CSCI 334
Stephen Freund

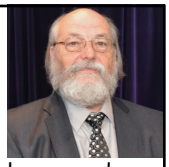
1

Principles of Security Design

- Ensure: secrecy, integrity, availability
- Least Privilege [Saltzer & Schroeder 75]
- Small Trusted Computing Base (TCB)


2

Trojan Horse [Ken Thompson]



- Create compiler binary that:
  A. inserts Trojan horse code to steal passwords when source code for login program is compiled
  B. inserts code to do (A) and (B) whenever source code for compiler is compiled
- Source code for login and compiler have no signs of attack
- Recompiling doesn't help...

3

Morris Worm [Robert Morris, Jr]

- Program that spreads across Internet and "lives forever"
  - Find new computers on Internet
  - Break into them
  - Copy source code, compile it, start running it, and hide tracks.
- Breaking in:
  - password guessing
  - buffer overrun in fingerd
  - bug in sendmail

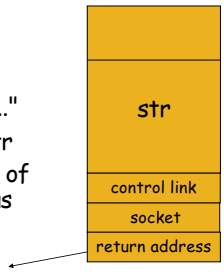
http://en.wikipedia.org/wiki/Robert_Tappan_Morris

4

Stack Smashing

```
void readName(FILE *socket) {
  char str[512];
  fgets(socket, str);
}
```

- Send name "**asd32423098c0sedh1...**"
- Overwrite past end of str
- Replace RA with address of code to perform malicious operation

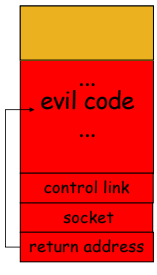


5

Stack Smashing

```
void readName(FILE *socket) {
  char str[512];
  fgets(socket, str);
}
```

- Send name "**asd32423098c0sedh1...**"
- Overwrite past end of str
- Replace RA with address of code to perform malicious operation



6

Context Has Changed

• TCB	Small
• Connections	Isolated
• Sys Admins	Skilled
• Vendors	Few
• Delivery	Physical
• Update freq.	Seldom
• Update size	Whole
• Executables	Large Apps
• Sensitive Data	Local HD

7

Stakes are High

- Personal:
  - privacy, identity theft, ransom-ware
- Companies/Governments…
  - Sony
  - Anonymous: RIA/MPAA, ISIS, …
  - PRISM: NSA breaches Google, Facebook, Apple, …
  - Stuxnet
  - North Korea missile program
  - 2016/2018/2020 US Election…
  - Bitcoin, Etherium, …
- Williams/CERT attacks?

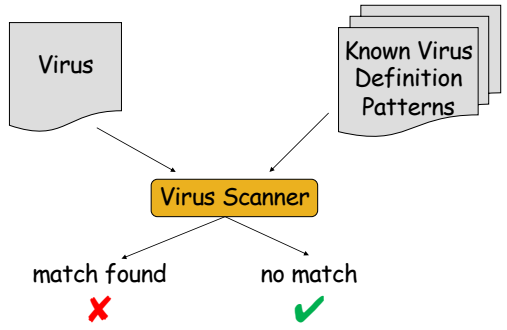
8

How to Prevent Attacks From Active Content

- Attach Social Stigma
- Educate Users
- Disable active content, disconnect from Web
- Virus Scanning
- Code Signing

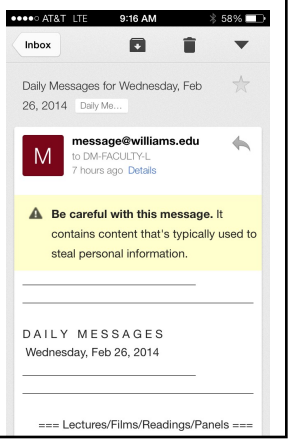
9

Virus Scanning



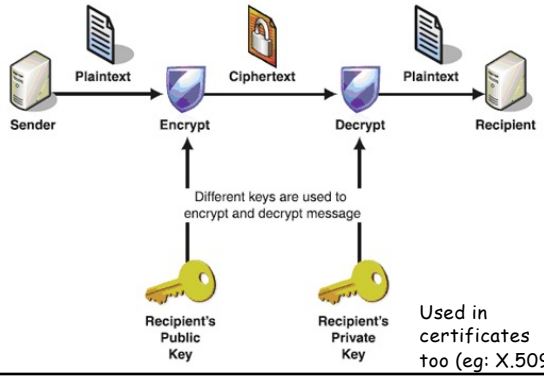
10

- Norton Antivirus: http://us.norton.com/antivirus/



11

Public Key Cryptography for Communication



Used in certificates too (eg: X.509)

12

Springer's private key

Code Signing

Program M

Encrypt

E(M,priv)

(certified program)

E(M,priv)
Program M

Steve's Inbox

13

Signature Checking

Springer's public key

#
Program M

Decrypt

Program M'=D(#,pub)

- Does M == M'?
  - ✔: Springer signed the code
  - ✘: Springer did not sign the code

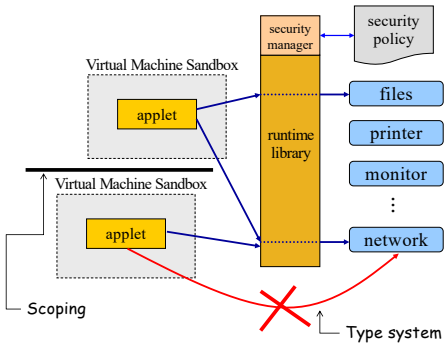
14

How to Prevent Attacks

- Attach Social Stigma
- Educate Users
- Disable active content, disconnect from Web
- Virus Scanning
- Code Signing
- **Language-Based Least Privilege**

15

Sandbox Security Model



16

Granting Privileges to Principals

- Local security policy file: `java.policy`

```
grant CodeBase "www.cs.williams.edu" {
    permission java.io.FilePermission
               "/home/data"
               "read", "write"
}

grant CodeBase "www.sneaky.com" {
    permission java.io.FilePermission
               "/tmp"
                "read", "write"
}
```

17

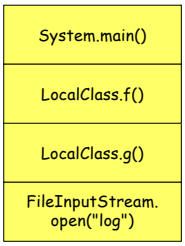
Security Manager

- Methods
  - checkRead
  - checkWrite
  - checkListen
  - checkConnect
  - ...
- Run-time system calls these methods prior to every resource access.

18

Stack Inspection

- Permission depends on:
  - calling method (based on principals)
  - all methods above it on stack

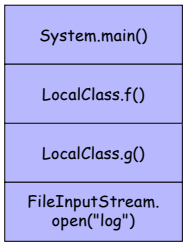



```
void open(String s) {
  SecurityManager.checkRead();
  ...
}
```

19

Stack Inspection

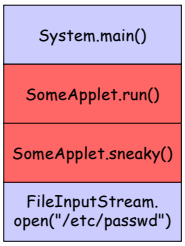
- Permission depends on:
  - calling method (based on principals)
  - all methods above it on stack
- Two Basic principals:
  - SYSTEM
  - UNTRUSTED



20

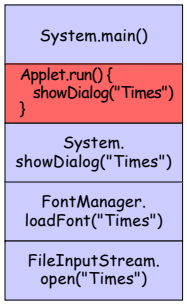
Stack Inspection

- Permission depends on:
  - calling method (based on principals)
  - all methods above it on stack
- Two Basic principals:
  - SYSTEM
  - UNTRUSTED



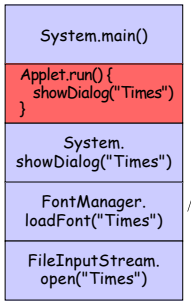
21

Stack Inspection (Example 2)



22

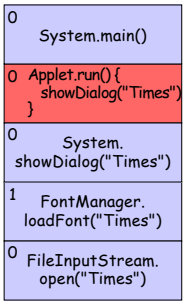
Stack Inspection (Example 2)




```
public Font loadFont(String s) {
  setPriviledged();
  ...
  input.open(s);
}
```

23

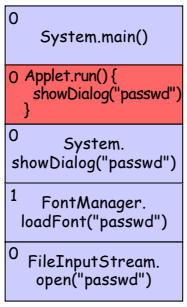
Stack Inspection (Example 2)




```
public Font loadFont(String s) {
  setPriviledged();
  ...
  input.open(s);
}
```

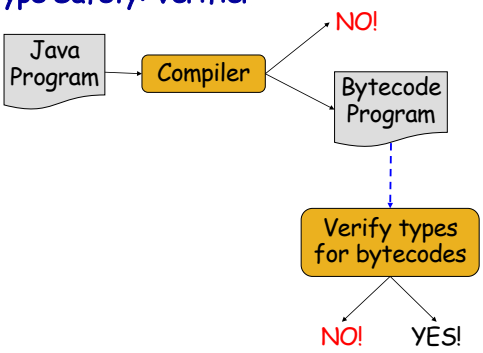
24

Stack Inspection (Example 2)



25

Type Safety: Verifier



26

Java Bytecodes

- Java:

```
class A extends Object {
  int i;
  void f(int val) { i = val + 1; }
}
```

- Bytecode:

```
Method void f(int)
  0 aload 0
  1 iload 1
  2 iconst 1
  3 iadd
  4 putfield #4 <Field int i>
  5 return
```

Does stack top have two integers?

Does field i exist for object on stack?

27

JavaScript WebAssembly!

Java Bytecodes

- Java:

```
class A extends Object {
  int i;
  void f(int val) { i = val + 1; }
}
```

- Bytecode:

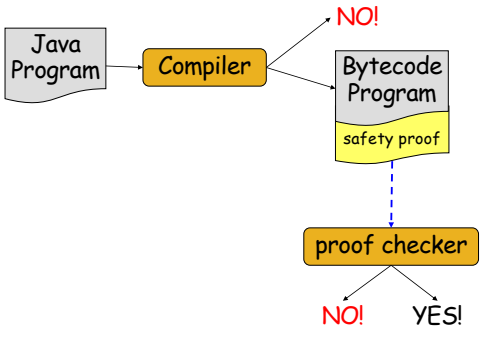
```
Method void f(int)
  0 aload 0
  1 iload 1
  2 iconst 1
  3 iadd
  4 putfield #4 <Field int i>
  5 return
```

S3=int::int::α
S4=int::α

S4=int::A::β
A has field "int i"
S5=β

28

Proof-Carrying Code



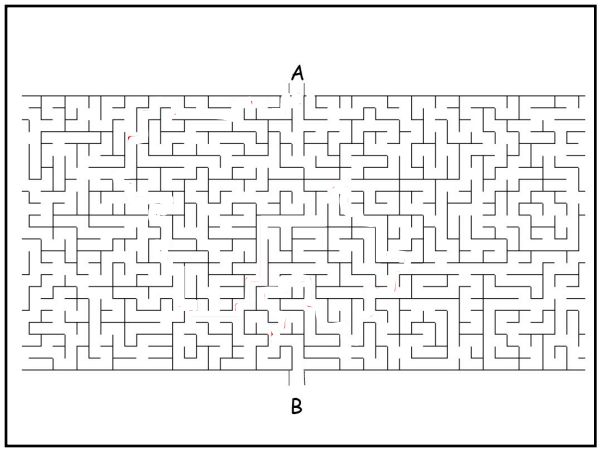
29

Java Bytecodes

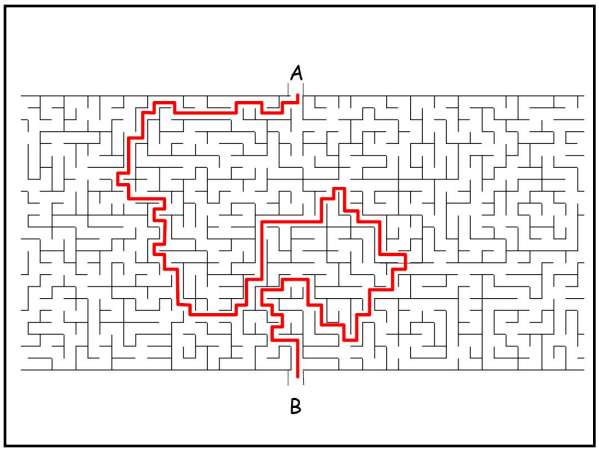
"Proof:"

```
Method void f(int)
  0 aload 0                    S=[]                    R0=A  R1=int
  1 iload 1                    S=A::[]                 R0=A  R1=int
  2 iconst 1                   S=int::A::[]            R0=A  R1=int
  3 iadd                       S=int::int::A::[]       R0=A  R1=int
  4 putfield #4 <Field int i>  S=int::A::[]            R0=A  R1=int
  5 return                     S=[]                    R0=A  R1=int
```

30



31



32

Post-Mortem

- Audit logs
- Look for anomalous behavior
- Reconstruct failures, prevent in future

- OpenSSL attack (2014?)
  - buffer overrun
  - divulged information from server
  - no audit log --- can't tell who was actually hacked...

33