Three Papers on Security

Phil Gibbons

15-712 F15

Lecture 24

Today's Reminders

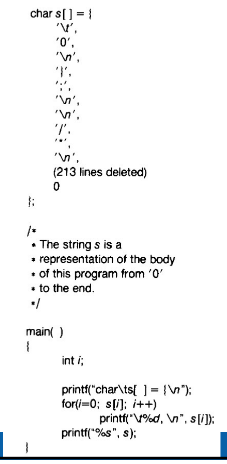
- **Wednesday: Highlights from SOSP'15 History Day**
  - No assigned readings. No summary due
  - Follow link in syllabus to see slides from all the talks
- **Wednesday's Office Hours will be 5-5:30 pm**
  - Shortened due to conflict with CSD Faculty meeting

Reflections on Trusting Trust
[CACM 1984]

- **Ken Thompson** (Google)
  - 1983 Turing Award lecture
  - 1980 NAE
  - 1990 Hamming Award
  - 1998 National Medal of Technology

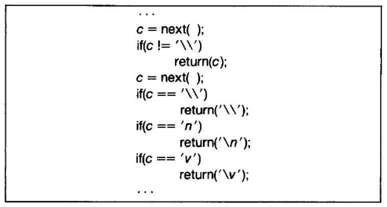
"The paper demonstrated that to have trust in a program, one cannot just rely on trust in the person who wrote it, or even on verifying the source code. One must also ensure that the entire tool chain used to produce and execute binaries is trustworthy."
– SigOps HoF citation (2011)

A Self-Reproducing Program



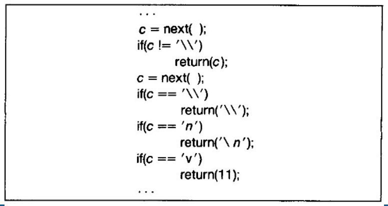
```
char s[ ] = {
        '\t',
        '0',
        '\n',
        '}',
        ';',
        '\n',
        '\n',
        '/',
        '*',
        '\n',
        (213 lines deleted)
        0
};

/*
 * The string s is a
 * representation of the body
 * of this program from '0'
 * to the end.
 */

main( )
{
        int i;

        printf("char\ts[ ] = {\n");
        for(i=0; s[i]; i++)
                printf("\t%d, \n", s[i]);
        printf("%s", s);
}
```

Interpreting Character Escape Chars



```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return('\v');
...
```

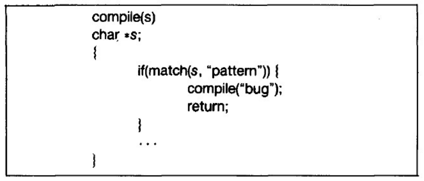
FIGURE 2.1.



```
...
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return(11);
...
```

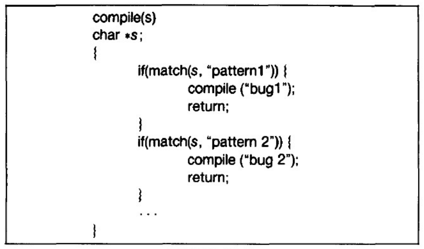
FIGURE 2.3.

Compile a Line of Source Code



```
compile(s)
char *s;
{
        if(match(s, "pattern")) {
                compile("bug");
                return;
        }
        ...
}
```

FIGURE 3.2.



```
compile(s)
char *s;
{
        if(match(s, "pattern1")) {
                compile ("bug1");
                return;
        }
        if(match(s, "pattern 2")) {
                compile ("bug 2");
                return;
        }
        ...
}
```

FIGURE 3.3.

bug 2 is self-reproducing program that inserts both Trojan Horses into the compiler

Can remove the bugs from the source of the compiler & the new binary will reinsert the bugs whenever it is compiled

Moral

"To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software."

"You can't trust code that you did not totally create yourself…No amount of source-level verification or scrutiny will protect you from using untrusted code."

"The act of breaking into a computer system has to have the same social stigma as breaking into a neighbor's house. It should not matter that the neighbor's door is unlocked."

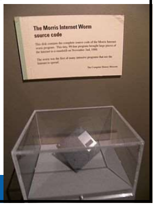
Crisis and Aftermath

[CACM 1989]

- **Eugene Spafford** (Purdue)
  - ACM Fellow 1997
  - AAAS Fellow 1999
  - IEEE Fellow 2000
  - Cybersecurity Hall of Fame 2013
- **First 2 papers he published were on this Worm**

Morris Worm

- **First Internet worm to gain widespread media attention**
  - Released on 11/2/98 by Robert Morris (Cornell grad student)
  - 2000 computers infected in 15 hours (dead until disinfected)
- **Worm contained no code to explicitly damage any system on which it ran**
- **But a computer could be infected multiple times, and all the additional processes would crash the computer**
  - 1 in 7 times (re-)infect a machine
  - Code had no mechanism to halt its spread
- **Internet was partitioned for several days, while regional networks disconnected from each other to avoid recontamination**



Morris Worm

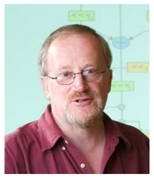
- **U.S. Government Accountability Office estimates damage at $100,000 to $10,000,000**
- **First felony conviction in U.S. under the 1986 Computer Fraud & Abuse Act**
  - 3 years probation, 400 hours of community service, $10,000 fine plus cost of supervision
- **Led to establishment of CERT at CMU**
- **Robert Morris now a Professor at MIT**
  - ACM Fellow

Morris Worm

- **Exploited known vulnerabilities in UNIX sendmail, finger, rsh/rexec and weak passwords**
- **Big debate in academic/industry/government circles as to suitable punishment**
  - Motivation: “Impress Jodie Foster?”
  - Can’t afford to tolerate the romanticization of computer vandals & computer criminals
- **Nature of Internet & Unix helped spread & defeat worm**
  - Shouldn’t increase obstacles to communication

Why Cryptosystems Fail
[CACM 1994]

- **Ross Anderson** (Cambridge University)
  - Fellow of the Royal Society (FRS)
  - Fellow of the Royal Academy of Engineering (FREng)



Based on a longer paper on CCS’93

Lack of Visibility into Failures

- **Lack of feedback about how fails in practice**
  - Compare with commercial airline crashes
  - False threat models
- **Cryptography in ATMs since early 1970s**
  - Of the hundreds of documented failures of ATM security, only two were due to a skilled (cryptology) attack

Main Causes of Phantom Withdrawals

- **Program Bugs**
  - 1 in 34,000 transactions are wrongly processed
- **Postal Intercept of ATM Cards**
  - E.g., 4000 Cambridge students open bank accounts every October; their ATM cards and PINS are delivered to college pigeonholes
- **Thefts by Bank Staff**
  - British banks dismiss 1% of their staff every year for disciplinary reasons, many for petty theft

More Exotic Attacks

- **Observe PINs being typed in & get account number from discarded receipts**
- **False ATM Terminals**
- **Programmer arranged for only 3 different PINs**
- **Etc**

Other Challenges

- **Outsourced or Frequently reorganized security teams**
- **Security products designed for high-tech attacks**
- **Ignoring the most important problem of how security features are embedded in real systems**
- **Security products should only be certified if they are simple enough for ordinary technical staff to use**
- **Unskilled implementers & multiple simultaneous threats**

Almost all security failures are due to implementation & management errors

Make Security Systems Robust

- Bridge builders use overdesign
- Aircraft engineers use extensive redundancy
- Neither suffices for security

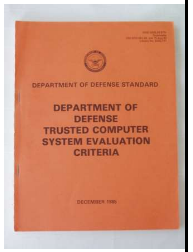
Explicitness should be the organizing principle for security robustness

Principles of Safety Critical Systems

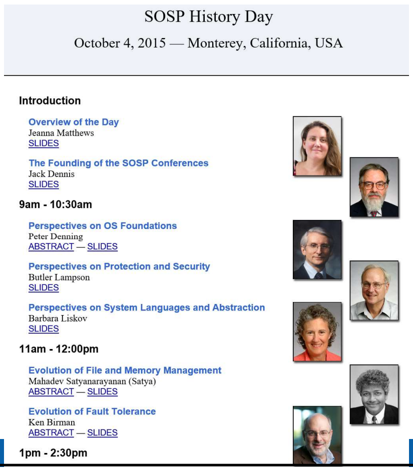
- Specification lists all possible failure modes
  - Every substantially new accident/incident ever reported
- Explain what strategy has been adopted to prevent (or make acceptably unlikely) each of these failure modes
- Spell out how each strategy was implemented, including the consequences when each single component fails
  - Technical factors, training, management issues
- Certification program reviewed by independent experts; test whether the equipment can in fact be operated by people with the stated level of skill & experience
  - Include monitoring program for reporting all incidents

Conclusions

- Component-level certification is unlikely to achieve its stated goals
- Future security standards should take much more account of system & human factors
- Bulk of security R&D budget is for activities that are of marginal relevance to real needs
- There are no "secure" systems; merely systems whose goals include beating foreign armies, preventing fraud, winning lawsuits---Must make goals explicit.


DEPARTMENT OF DEFENSE STANDARD
DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA
DECEMBER 1985

Wednesday's Class


SOSP History Day

October 4, 2015 — Monterey, California, USA

Introduction

Overview of the Day
Jeanna Matthews
SLIDES

The Founding of the SOSP Conferences
Jack Dennis
SLIDES

9am - 10:30am

Perspectives on OS Foundations
Peter Denning
ABSTRACT — SLIDES

Perspectives on Protection and Security
Butler Lampson
SLIDES

Perspectives on System Languages and Abstraction
Barbara Liskov
SLIDES

11am - 12:00pm

Evolution of File and Memory Management
Mahadev Satyanarayanan (Satya)
ABSTRACT — SLIDES

Evolution of Fault Tolerance
Ken Birman
ABSTRACT — SLIDES

1pm - 2:30pm