Lecture 18
November 6, 2024

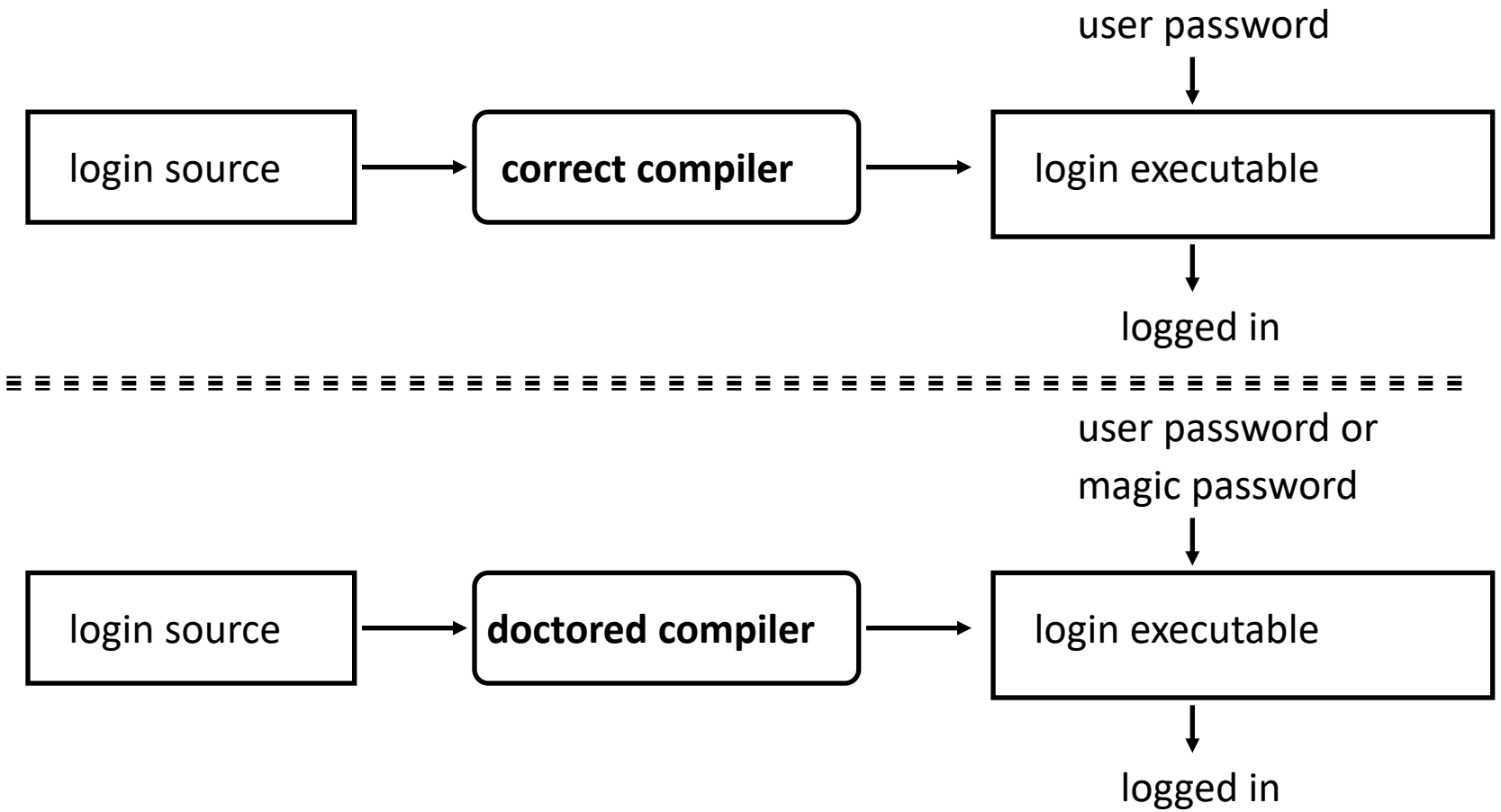
Replicating Trojan Horse

- Trojan horse that makes copies of itself
  - Also called *propagating Trojan horse*
  - Early version of *animal* game used this to delete copies of itself
- Hard to detect
  - 1976: Karger and Schell suggested modifying compiler to include Trojan horse that copied itself into specific programs including later version of the compiler
  - 1980s: Thompson implements this

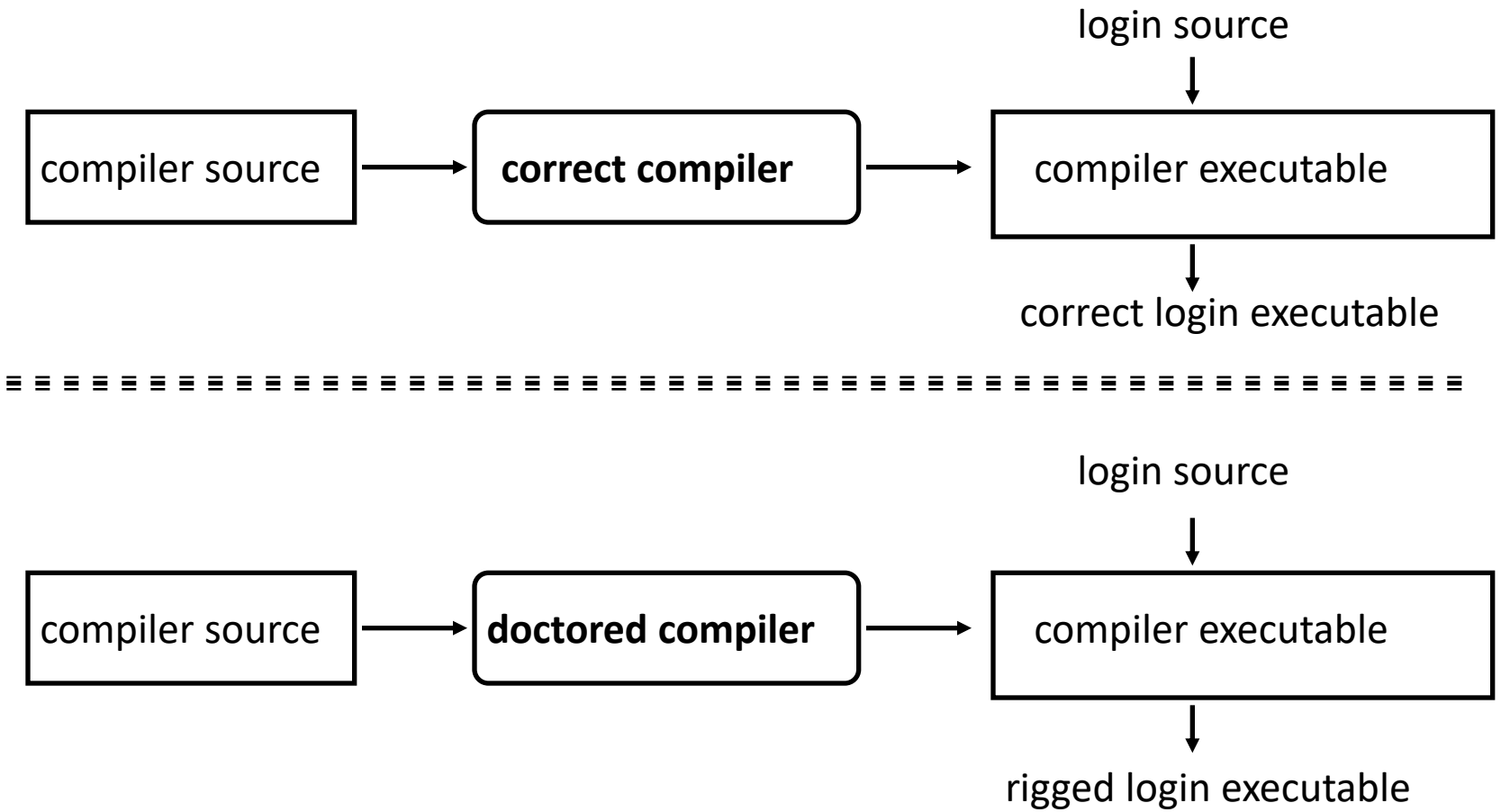
Thompson's Compiler

- Modify the compiler so that when it compiles *login*, *login* accepts the user's correct password or a fixed password (the same one for all users)
- Then modify the compiler again, so when it compiles a new version of the compiler, the extra code to do the first step is automatically inserted
- Recompile the compiler
- Delete the source containing the modification and put the undoctored source back

The *login* Program



The Compiler



Comments

- Great pains taken to ensure second version of compiler never released
  - Finally deleted when a new compiler executable from a different system overwrote the doctored compiler
- The point: *no amount of source-level verification or scrutiny will protect you from using untrusted code*
  - Also: having source code helps, but does not ensure you're safe

Computer Virus

- Program that inserts itself into one or more files and performs some action
  - *Insertion phase* is inserting itself into file
  - *Execution phase* is performing some (possibly null) action
- Insertion phase *must* be present
  - Need not always be executed
  - Lehigh virus inserted itself into boot file only if boot file not infected

Pseudocode

```
beginvirus:
  if spread-condition then begin
    for some set of target files do begin
      if target is not infected then begin
        determine where to place virus instructions
        copy instructions from beginvirus to endvirus
          into target
        alter target to execute added instructions
      end;
    end;
  end;
  perform some action(s)
  goto beginning of infected program
endvirus:
```

Trojan Horse Or Not?

- Yes
  - Overt action = infected program's actions
  - Covert action = virus' actions (infect, execute)
- No
  - Overt purpose = virus' actions (infect, execute)
  - Covert purpose = none
- Semantic, philosophical differences
  - Defenses against Trojan horse also inhibit computer viruses

History

- Programmers for Apple II wrote some
  - Not called viruses; very experimental
- Fred Cohen
  - Graduate student who described them
  - Teacher (Adleman, of RSA fame) named it “computer virus”
  - Tested idea on UNIX systems and UNIVAC 1108 system

Cohen's Experiments

- UNIX systems: goal was to get superuser privileges
  - Max time 60m, min time 5m, average 30m
  - Virus small, so no degrading of response time
  - Virus tagged, so it could be removed quickly
- UNIVAC 1108 system: goal was to spread
  - Implemented simple security property of Bell-LaPadula
  - As writing not inhibited (no *-property enforcement), viruses spread easily

First Reports of Viruses in the Wild

- Brain (Pakistani) virus (1986)
  - Written for IBM PCs
  - Alters boot sectors of floppies, spreads to other floppies
- MacMag Peace virus (1987)
  - Written for Macintosh
  - Prints “universal message of peace” on March 2, 1988 and deletes itself

More Reports

- Duff's experiments (1987)
  - Small virus placed on UNIX system, spread to 46 systems in 8 days
  - Wrote a Bourne shell script virus
- Highland's Lotus 1-2-3 virus (1989)
  - Stored as a set of commands in a spreadsheet and loaded when spreadsheet opened
  - Changed a value in a specific row, column and spread to other files

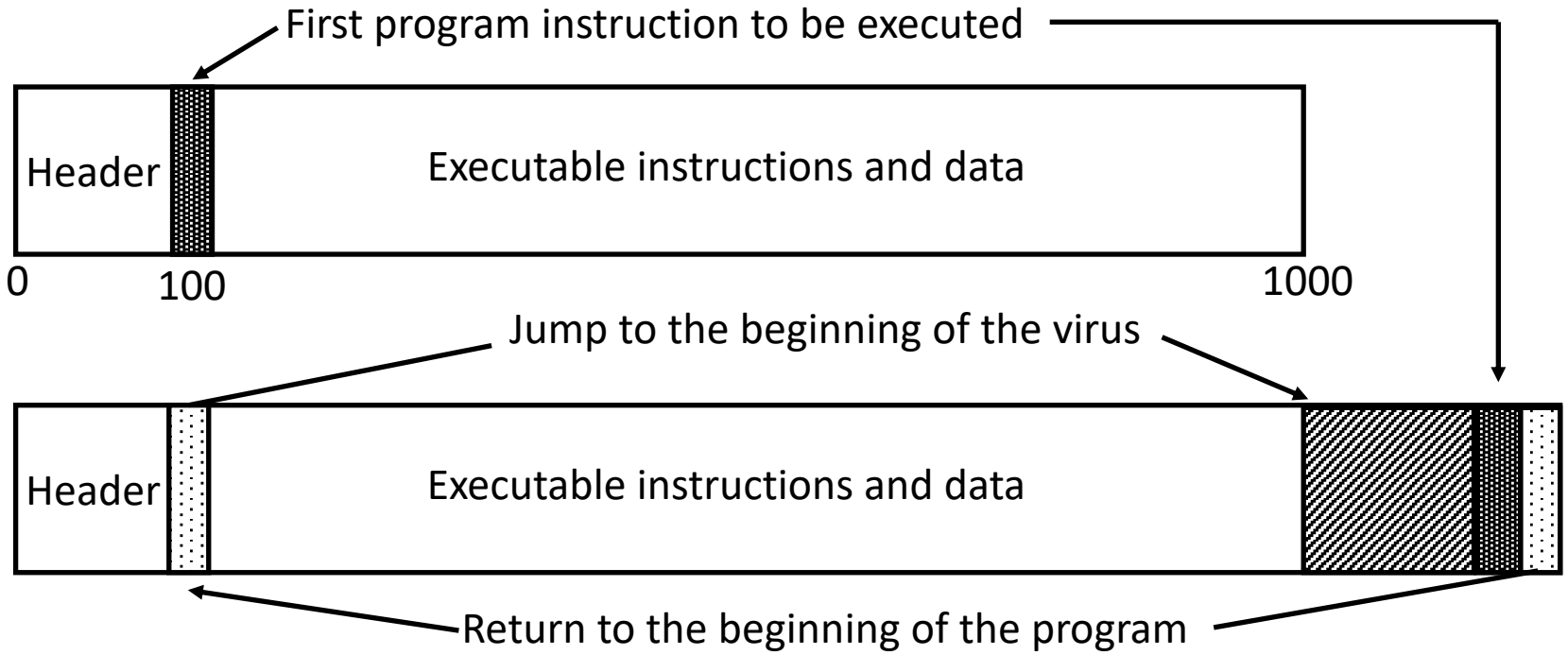
Infection Vectors

- Boot sector infectors
- Executable infectors
- Data infectors
- These are not mutually exclusive; some viruses do two or three of these

Boot Sector Infectors

- A virus that inserts itself into the boot sector of a disk
  - Section of disk containing code
  - Executed when system first “sees” the disk
    - Including at boot time …
- Example: Brain virus
  - Moves disk interrupt vector from 13H to 6DH
  - Sets new interrupt vector to invoke Brain virus
  - When new floppy seen, check for 1234H at location 4
    - If not there, copies itself onto disk after saving original boot block; if no free space, doesn’t infect but if any free space, it infects, possibly overwriting used disk space
    - If there, jumps to vector at 6DH

Executable Infectors



- A virus that infects executable programs
  - Can infect either .EXE or .COM on PCs
  - May append itself (as shown) or put itself anywhere, fixing up binary so it is executed at some point

Executable Infectors (*con't*)

- Jerusalem (Israeli) virus
  - Checks if system infected
    - If not, set up to respond to requests to execute files
  - Checks date
    - If not 1987 or Friday 13th, set up to respond to clock interrupts and then run program
    - Otherwise, set destructive flag; will delete, not infect, files
  - Then: check all calls asking files to be executed
    - Do nothing for COMMAND.COM
    - Otherwise, infect or delete
  - Error: doesn't set signature when .EXE executes
    - So .EXE files continually reinfected

Macro Viruses

- A virus composed of a sequence of instructions that are interpreted rather than executed directly
- Can infect either executables (Duff's shell virus) or data files (Highland's Lotus 1-2-3 spreadsheet virus)
- Independent of machine architecture
  - But their effects may be machine dependent

Example

- Melissa
  - Infected Microsoft Word 97 and Word 98 documents
    - Windows and Macintosh systems
  - Invoked when program opens infected file
  - Installs itself as “open” macro and copies itself into Normal template
    - This way, infects any files that are opened in future
  - Invokes mail program, sends itself to everyone in user’s address book
    - Used a mail program that most Macintosh users didn’t use, so this was rare for Macintosh users

Multipartite Viruses

- A virus that can infect either boot sectors or executables
- Typically, two parts
  - One part boot sector infector
  - Other part executable infector

Concealment

- Terminate and stay resident (TSR)
- Stealth
- Encryption
- Polymorphism
- Metamorphism

TSR Viruses

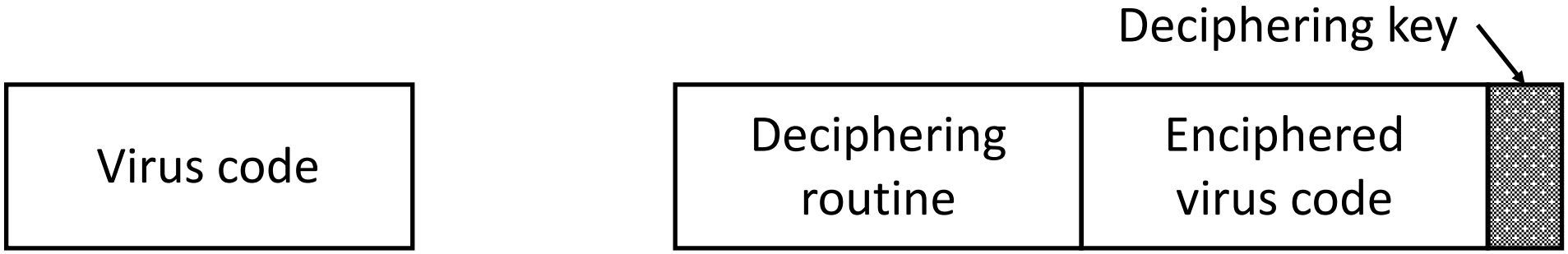
- A virus that stays active in memory after the application (or bootstrapping, or disk mounting) is completed
  - Non-TSR viruses only execute when host application executes
- Examples: Brain, Jerusalem viruses
  - Stay in memory after program or disk mount is completed

Stealth Viruses

- A virus that conceals infection of files
- Example: IDF (also called Stealth or 4096) virus modifies DOS service interrupt handler as follows:
  - Request for file length: return length of *uninfected* file
  - Request to open file: temporarily disinfect file, and reinfect on closing
  - Request to load file for execution: load infected file

Encrypted Viruses

- A virus that is enciphered except for a small deciphering routine
  - Detecting virus by signature now much harder as most of virus is enciphered



Example

```
(* Decryption code of the 1260 virus *)
(* initialize the registers with the keys *)
rA = k1;
rB = k2;
(* initialize rC with the virus; starts at sov, ends at eov *)
rC = sov;
(* the encipherment loop *)
while (rC != eov) do begin
        (* encipher the byte of the message *)
        (*rC) = (*rC) xor rA xor rB;
        (* advance all the counters *)
        rC = rC + 1;
        rA = rA + 1;
end
```

Polymorphic Viruses

- A virus that changes its form each time it inserts itself into another program
- Idea is to prevent signature detection by changing the “signature” or instructions used for deciphering routine
  - At instruction level: substitute instructions
  - At algorithm level: different algorithms to achieve the same purpose
- Toolkits to make these exist (Mutation Engine, Trident Polymorphic Engine)
- After decipherment, same virus loaded into memory
  - Virus is encrypted; decryption routine is obscured (polymorphicized?)

Example

- These are different instructions (with different bit patterns) but have the same effect:
  - add 0 to register
  - subtract 0 from register
  - xor 0 with register
  - no-op
- Polymorphic virus would pick randomly from among these instructions

Metamorphic

- Like polymorphic, but virus itself is also obscured
  - So two instances of virus would look different when loaded into memory
- When decrypted, virus may have:
  - Two completely different implementations
  - Two completely different algorithms producing same result

Example

- W95/Zmist virus distributes itself throughout code being infected
- On finding file to infect:
  - p = 0.1: insert jump instructions between each set of non-jump instructions
  - p = 0.1: infect file with unencrypted copy of Zmist
  - p = 0.8: if file has section with initialized data that is writeable, infect file with polymorphic encrypted version of Zmist; otherwise, infect file with unencrypted copy of Zmist
    - In first case, virus expands that section, inserts virus code as it is decrypted, and executes that code; decrypting code preserves registers so they can be restored
- On execution, allocates memory to put virus engine in; that creates new instance of (transformed) virus

Computer Worms

- A program that copies itself from one computer to another
- Origins: distributed computations
  - Schoch and Hupp: animations, broadcast messages
  - Segment: part of program copied onto workstation
  - Segment processes data, communicates with worm's controller
  - Any activity on workstation caused segment to shut down

Example: Internet Worm of 1988

- Targeted Berkeley, Sun UNIX systems
  - Used virus-like attack to inject instructions into running program and run them
  - To recover, had to disconnect system from Internet and reboot
  - To prevent re-infection, several critical programs had to be patched, recompiled, and reinstalled
- Analysts had to disassemble it to uncover function
- Disabled several thousand systems in 6 or so hours

Example: Christmas Worm

- Distributed in 1987, designed for IBM networks
- Electronic letter instructing recipient to save it and run it as a program
  - Drew Christmas tree, printed “Merry Christmas!”
  - Also checked address book, list of previously received email and sent copies to each address
- Shut down several IBM networks
- Really, a macro worm
  - Written in a command language that was interpreted

Computer Worm Structure

- *Target Selection*: worm determines which systems to spread to
- *Propagation*: worm attempts to infect chosen targets
- *Execution*: worm carries out action after it becomes resident on a target
  - This phase may be empty

Example: Internet Worm

- Target selection: chose targets from lists of trusted hosts, and hosts trusted by users whose passwords had been guessed
- Propagation: tried to exploit 4 vulnerabilities
  - *sendmail* (SMTP server) in debug mode
  - *fingerd* (information server) buffer overflow attack
  - used guessed passwords
  - tried to exploit trust relationships
- Execution: took actions to:
  - Concealed its presence
  - Prevent reinfection
  - tried to guess passwords on local system (to be used in target selection)

Stuxnet

- Found in 2010, targeted Siemens centrifuges used in process to enrich uranium
  - Compromised Windows software first, then the PLC in centrifuges
    - Very sophisticated evasion, exploits, and use of first PLC rootkit
  - Spun them at nonstandard speeds so they tore apart
- Entered system via infected USB stick with a Trojan horse
  - Looked on local network for Windows-based systems to infect; if found, infected no more than 3
- On system, checked to see if it was part of a specific industrial control system
  - No: did nothing
  - Ye: acted

Stuxnet (*con't*)

- Tried to download most current version of itself
- Exploited vulnerabilities in infected system's PLC to take control of attached centrifuges
  - Also corrupted information sent to the controllers so they would not detect anything was wrong until centrifuges went out of control
- Believed developed by one or more nation-states due to its complexity, sophistication
- Other equally sophisticated worms found since then
  - Flame: spread in ways similar to Stuxnet, but only gathers information from microphones, keystrokes, network traffic, and so forth for the attackers to retrieve

Importance of Stuxnet

- Earlier research showed physical systems vulnerable to attacks from connected computers
- Stuxnet showed these attacks can be launched over the Internet

Bots

- *bot*: malware that carries out some action in co-ordination with other bots
- *botmaster*: attacker controlling the bots on one or more systems
- *command and control (C&C) server*, *mothership*: system(s) the attacker uses to control the bots
- *C&C channels*: communication paths used to communicate with bots
  - Distinguishing characteristic of bot is the use of this channel
  - Can be triggered, updated over this
- *botnet*: a collection of bots

Life Cycle of a Bot in a Botnet

1. Bot infects system
2. Bot checks for a network connection, looks for either C&C server or another bot it can communicate with
3. Bot gets commands sent by C&C server or other bot
   - These may include adding components to add to what the bot can do
4. Bot executes these commands
   - May send results to somewhere else