

# Computer Security DD2395

<http://www.csc.kth.se/utbildning/kth/kurser/DD2395/dasakh11/>

Fall 2011

Sonja Buchegger

[buc@kth.se](mailto:buc@kth.se)

Lecture 7

Malicious Software

# Course Admin

- Lab 2:
  - prepare before lab session
  - signup!
- Lab 3:
  - prepare: webgoat, gruyere
- Lab 4:
  - signup
  - finding group partners: meet here during break

# Malicious Software

- programs exploiting system vulnerabilities
- known as malicious software or malware
  - program fragments that need a host program
    - e.g. viruses, logic bombs, and backdoors
  - independent self-contained programs
    - e.g. worms, bots
  - replicating or not
- sophisticated threat to computer systems

# Malware Terminology

- Virus
- Worm
- Logic bomb
- Trojan horse
- Backdoor (trapdoor)
- Mobile code
- Auto-rooter Kit (virus generator)
- Spammer and Flooder programs
- Keyloggers, Spyware
- Rootkit
- Zombie, bot
- Adware

# Would you trust this program?



# Trojan Horse

- First identified at NSA in 1972 by Daniel Edwards
- It's a program with two purposes, one obvious and one hidden from the user
- Today it's often used to install other software or backdoors
- Trojan horses can be built from existing programs using a special wrapper
- Or designed from the start to be one.

# What would you do?

- How to get someone to run a trojan?
- How to not run a trojan?

# Backdoor

- Software that gives access to a system
- Bypassing OS restrictions
- Can be part of a trojan
- Often installed for legitimate reasons
- Only to later be abused
- Typically very very hard to find

# Legitimate Reasons?

- What would be a legitimate reason to install a backdoor?

# Grayware

- In the gray zone between harmless and harmful, mostly annoying
- Popup windows
- For teh lulz
- Can include adware, spyware



# Logic Bomb

- A small bit of code that triggers on a specific condition
- Typically with malicious results
- No vector for spreading
- Installed directly

# Viruses

- piece of software that infects programs
  - modifying them to include a copy of the virus
  - so it executes secretly when host program is run
- specific to operating system and hardware
  - taking advantage of their details and weaknesses
- a typical virus goes through phases of:
  - dormant
  - propagation
  - triggering
  - execution

# Virus Structure

- components:
  - infection mechanism - enables replication
  - modification engine – for disguise
  - trigger - event that makes payload activate
  - payload - what it does, malicious or benign
- prepended / appended / embedded
- when infected program invoked, executes virus code then original program code
- can block initial infection (difficult)
- or propagation (with access controls)

# Virus Structure

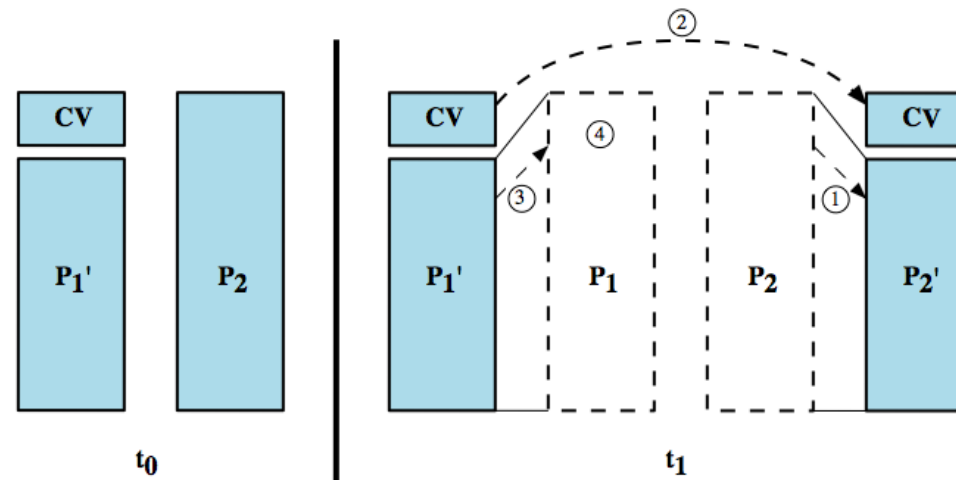
```
program V :=  
  
{goto main;  
 1234567;  
  
  subroutine infect-executable :=  
    {loop:  
      file := get-random-executable-file;  
      if (first-line-of-file = 1234567)  
        then goto loop  
        else prepend V to file; }  
  
  subroutine do-damage :=  
    {whatever damage is to be done}  
  
  subroutine trigger-pulled :=  
    {return true if some condition holds}  
  
main:  main-program :=  
  {infect-executable;  
  if trigger-pulled then do-damage;  
  goto next;}  
  
next:  
  
}
```

# Virus Classification

- boot sector
- file infector
- macro virus
- encrypted virus: different keys
- stealth virus: evade detection, e.g. compression
- polymorphic virus
- metamorphic virus

# Compression Virus

```
program CV :=  
{goto main;  
 01234567;  
  
  subroutine infect-executable :=  
    {loop:  
      file := get-random-executable-file;  
      if (first-line-of-file = 01234567) then goto loop;  
    (1)  compress file;  
    (2)  prepend CV to file;  
    }  
  
main:  main-program :=  
  {if ask-permission then infect-executable;  
  (3)  uncompress rest-of-file;  
  (4)  run uncompressed file;}  
}
```



# Polymorphic Virus

- A virus can take things one step further: Rebuild the whole virus at every infection to something functionally identical
- There are many ways to do nothing on a computer
- Instructions can be reordered in many ways
- To detect these the AV engine often has to simulate the virus to figure out what it is.

# Metamorphic Virus

- Complete rewrite
- Can also change behavior

# Macro Virus

- became very common in mid-1990s since
  - platform independent
  - infects documents
  - is easily spread
- exploit macro capability of office apps
  - executable program embedded in office doc
  - often a form of Basic
- more recent releases include protection
- recognized by many anti-virus programs

# E-Mail Viruses

- more recent development
- e.g. Melissa
  - exploits MS Word macro in attached doc
  - if attachment opened, macro activates
  - sends email to all on users address list
  - and does local damage
- then saw versions triggered reading email
- hence much faster propagation

# Virus Countermeasures

- prevention - ideal solution but difficult
- realistically need:
  - detection
  - identification
  - removal
- if detected but can't identify or remove, must discard and replace infected program

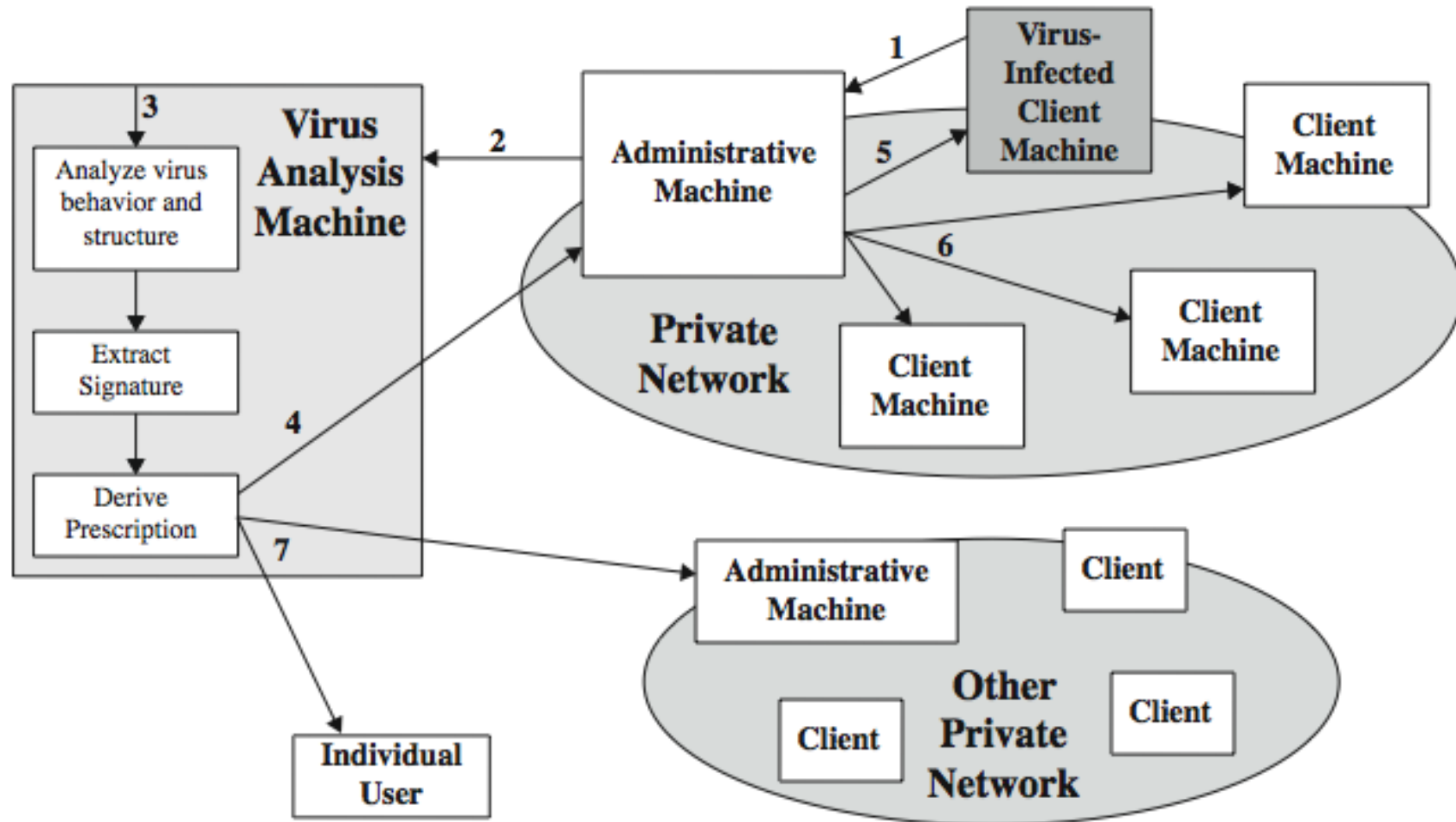
# Anti-Virus Evolution

- virus & antivirus tech have both evolved
- early viruses simple code, easily removed
- as become more complex, so must the countermeasures
- generations
  - first - signature scanners
  - second - heuristics
  - third - identify actions
  - fourth - combination packages

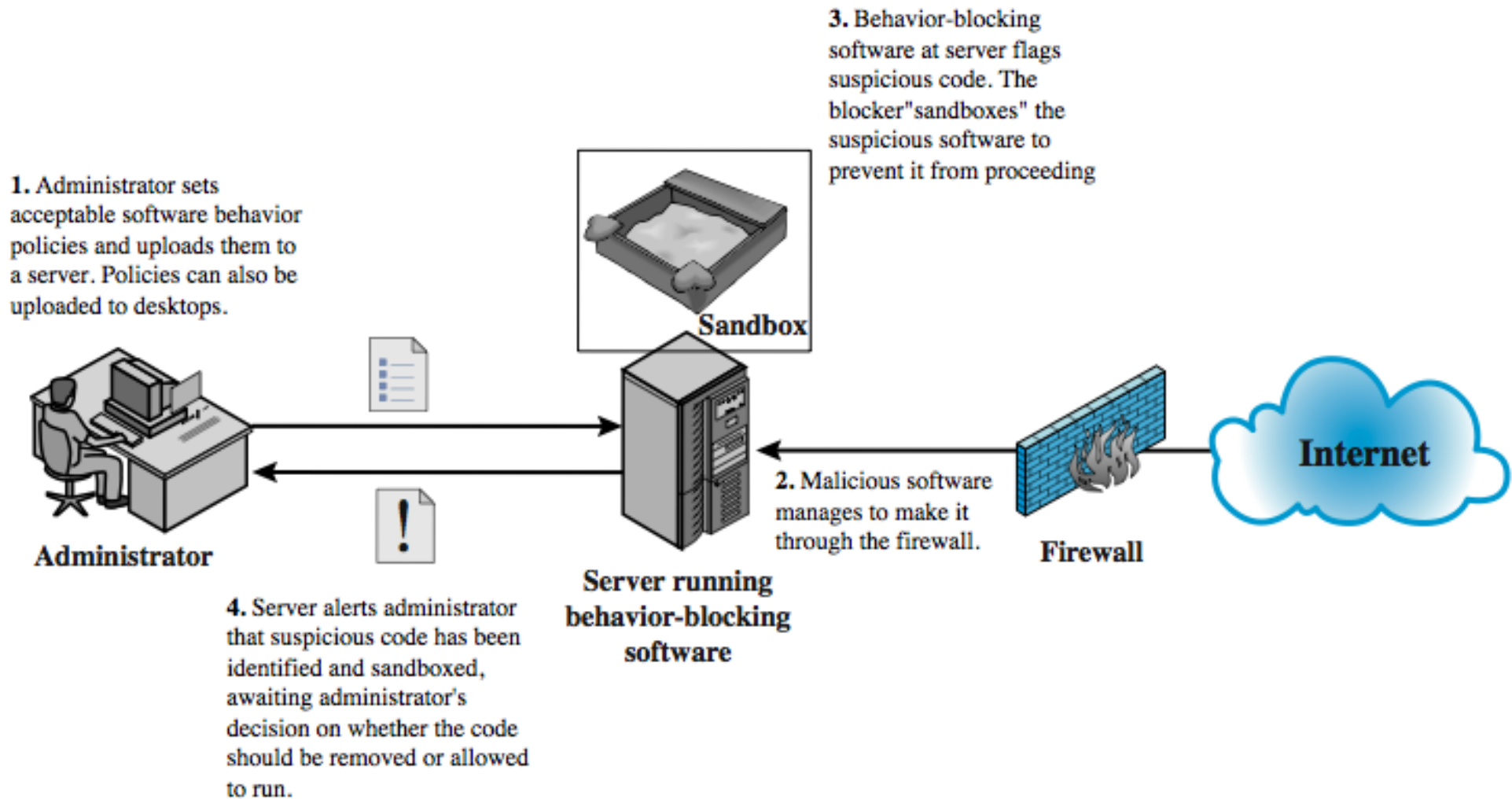
# Generic Decryption

- runs executable files through GD scanner:
  - CPU emulator to interpret instructions
  - virus scanner to check known virus signatures
  - emulation control module to manage process
- lets virus decrypt itself in interpreter
- periodically scan for virus signatures
- issue is long to interpret and scan
  - tradeoff chance of detection vs time delay

# Digital Immune System



# Behavior-Blocking Software



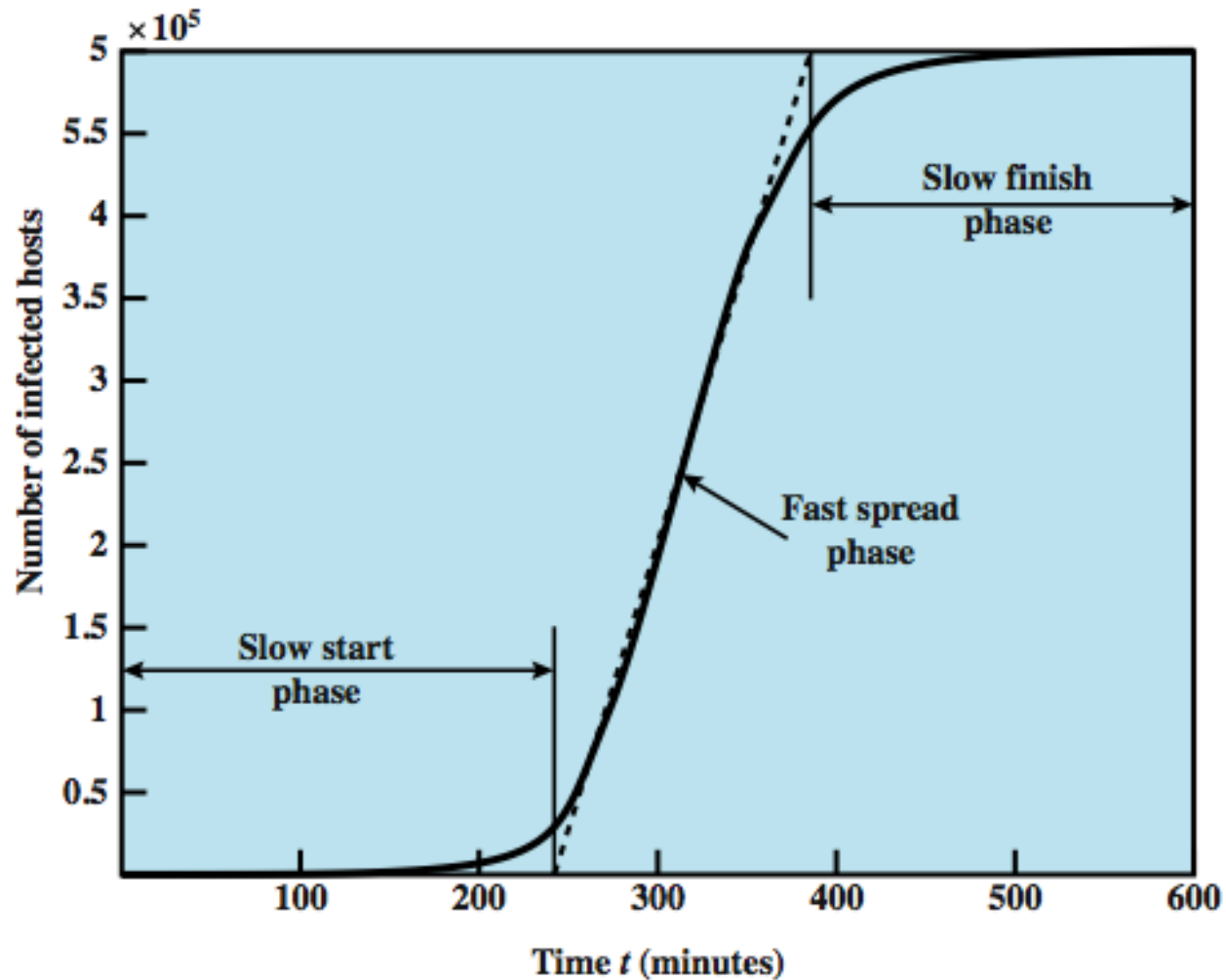
# Worms

- replicating program that propagates over net
  - using email, remote exec, remote login
- has phases like a virus:
  - dormant, propagation, triggering, execution
  - propagation phase: searches for other systems, connects to it, copies self to it and runs
- may disguise itself as a system process
- implemented by Xerox Palo Alto labs in 1980' s

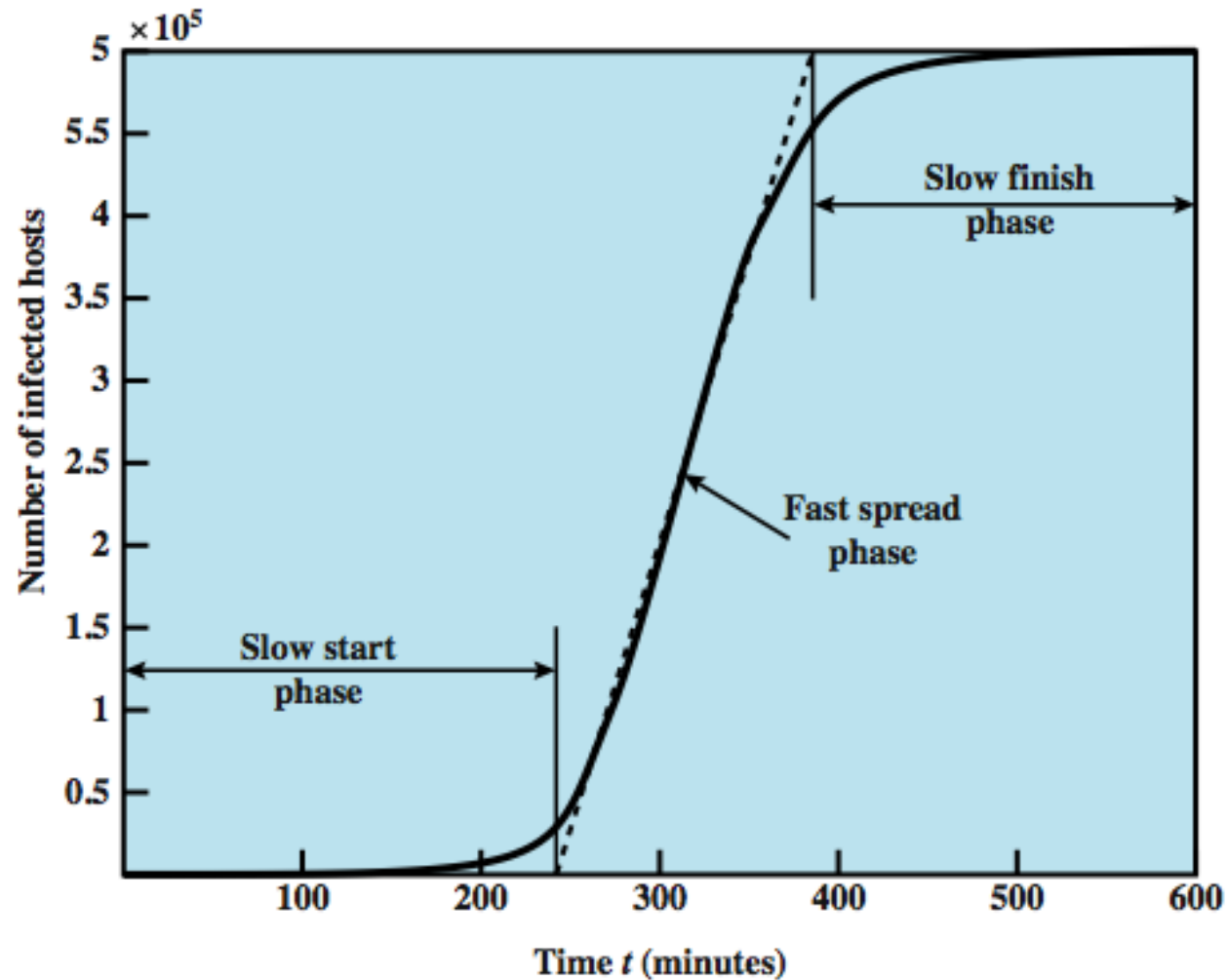
# Morris Worm

- one of best known early worms
- released by Robert Morris in 1988
- various attacks on UNIX systems
  - cracking password file to use login/password to logon to other systems
  - exploiting a bug in the finger protocol
  - exploiting a bug in sendmail
- if succeed have remote shell access
  - sent bootstrap program to copy worm over

# Worm Propagation Model



# Why the slow finish phase?



# Recent Worm Attacks

- Code Red
  - July 2001 exploiting MS IIS bug
  - probes random IP address, does DDoS attack
  - consumes significant net capacity when active
- Code Red II variant includes backdoor
- SQL Slammer
  - early 2003, attacks MS SQL Server
  - compact and very rapid spread
- Mydoom
  - mass-mailing e-mail worm that appeared in 2004
  - installed remote access backdoor in infected systems

# Recent Worm Attacks

- Conficker 2009
- Stuxnet 2010
- Duqu 2011

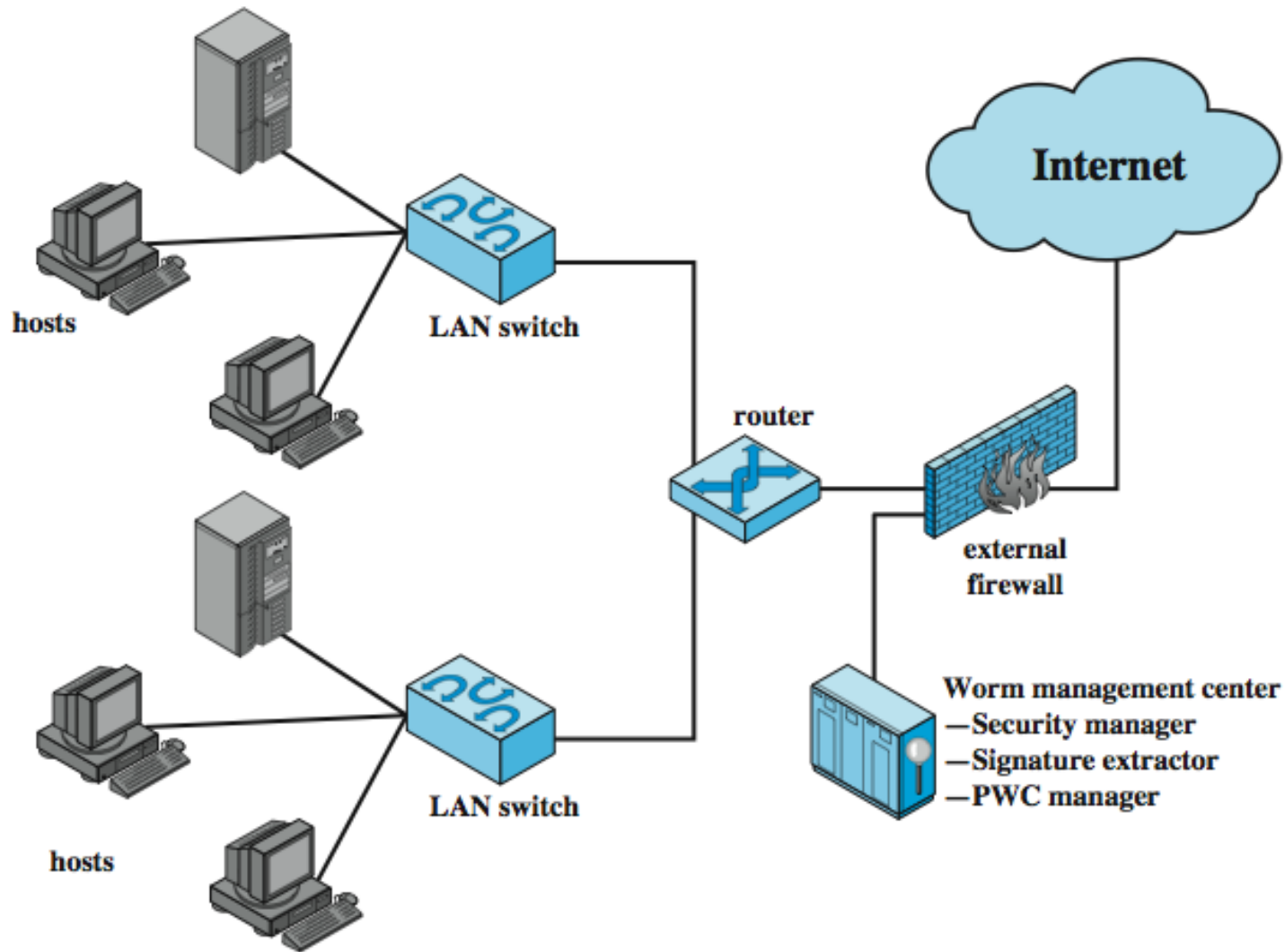
# Worm Technology

- multiplatform
- multi-exploit
- ultrafast spreading
- polymorphic
- metamorphic
- transport vehicles
- zero-day exploit

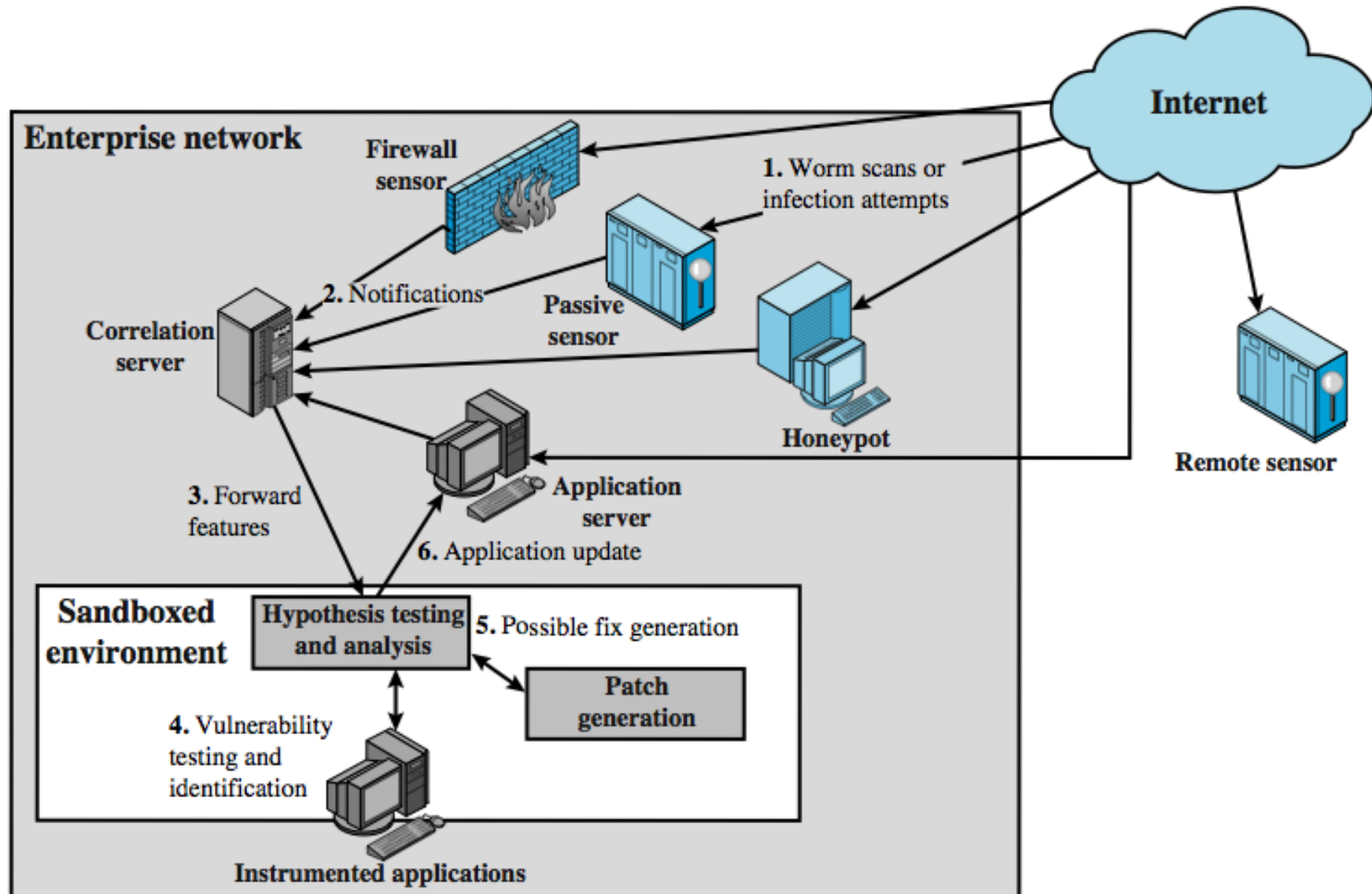
# Worm Countermeasures

- overlaps with anti-virus techniques
- once worm on system A/V can detect
- worms also cause significant net activity
- worm defense approaches include:
  - signature-based worm scan filtering
  - filter-based worm containment
  - payload-classification-based worm containment
  - threshold random walk scan detection
  - rate limiting and rate halting

# Proactive Worm Containment



# Network Based Worm Defense



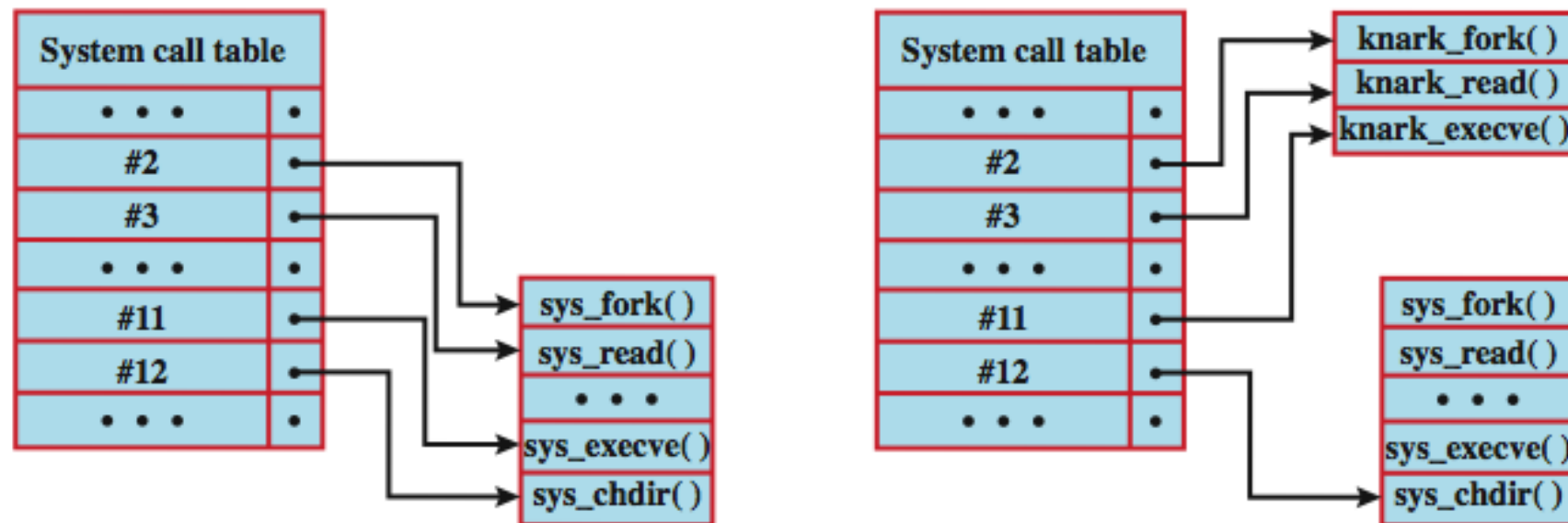
# Bots

- program taking over other computers
- to launch hard to trace attacks
- if coordinated form a botnet
- characteristics:
  - remote control facility
    - via IRC/HTTP etc
  - spreading mechanism
    - attack software, vulnerability, scanning strategy
- various counter-measures applicable

# Rootkits

- set of programs installed for admin access
- malicious and stealthy changes to host O/S
- may hide its existence
  - subverting report mechanisms on processes, files, registry entries etc
- may be:
  - persistent or memory-based
  - user or kernel mode
- installed by user via trojan or intruder on system
- range of countermeasures needed

# Rootkit System Table Mods



(a) Normal kernel memory layout

(b) After nkark install

# Summary

- introduced types of malicious software
  - incl backdoor, logic bomb, trojan horse
  - virus types and countermeasures
- worm types and countermeasures
- bots
- rootkits