**Lecture 8: Malicious Software III**

*Summary*

This is the last lecture on Malicious Software. We will discuss various payloads that may be carried and delivered by malware and approaches to protecting against malware.

*Payloads*

Reference: P&P, Sections 3.3 and 3.4

**System Damage**

Example: Chernobyl Virus (1998)

Example: CryptoLocker (2013)

**Rootkits**

Kernel-mode vs. user-mode

Function Hooking

Detecting

Example: Sony BMG Copy Protection (2005)

Example: AFX Windows Rootkit (2003)

**Botnets**

**Privacy Invasive Software**

*Protection Against Malware*

Reference: P&P Sections 3.3 and 3.5

**Deploying Systems**

Detect, Identify, Remove

Detection

Signature-based scanners

Heuristics-based scanners

Perimeter Scanners — Intrusion Detection Systems

“Best Pratices”

Diversity of Systems

Robustness of Software

Limit user privilege

Improve authentication

Monitor networks

Use malicious software detection and removal tools

Prepare for recovery

**Developing Systems — Software Engineering**

Design

Modularity

Encapsulation

Information Hiding

Mutual Suspicion

Peer Review

Review

Walk-through

Inspection

Other Techniques

Hazard Analysis

Testing

Static Analysis

Configuration Management

*Exercises*

P&P, Chapter 3, Exercises 11, 12, 15.

Suppose we have a program *D* that is claimed to be able to determine whether a given input program *P* is a virus or not, returning “True” if *P* is a virus and “False” if it is not. Consider the following program:

 Program V {

 main {

 if D(V) then

 goto next;

 else

 infect-executable;

 }

next:

}

The module infect-executable searches the system for executable programs that can be infected and replicates the program *V* to all such programs. Can *D* decide whether or not *V* is a virus?

You find a USB stick in a campus parking lot. To what threats might you expose your computer if you plug-in the USB stick? What steps could you take to mitigate the threats?