
Reverse Code Engineering: An In-Depth Analysis of the Bagle Virus

Konstantin Rozinov
(krozinov@yahoo.com)

Bell Labs
Government Communications Laboratory

Overview

- Introduction
 - *To myself and the project*
- Basic x86 Concepts
 - *Registers, assembly, runtime data structures*
- Virus Overview
 - *Definitions*
- Bagle Virus Disassembly
 - *Resources, approach, problems and solutions*
- Conclusions
 - *Results and future work*
- Demo

Introduction

- A little background information about myself:
 - My name is Konstantin Rozinov.
 - I attend Polytechnic University in Brooklyn, New York.
 - I am finishing up my masters in CS.
 - I will graduate in Spring 2005.
- My summer internship at Lucent:
 - I was part of the Internet Research Department headed by Tom Reddington.
 - My manager was Ted Wroblecka, head of the Systems and Software Group.
 - My mentor was Bjoern Luettmann.

Introduction *(cont'd)*

- Project Description:

The goal of this project is to try to answer the following three questions:

- *How do you reverse engineer a virus?*
- *Can reverse engineering a virus lead to better ways of detecting, preventing, and recovering from a virus and its future variants?*
- *Can reverse engineering be done more efficiently?*

Basic x86 Concepts

- You should be familiar with:
 - Registers
 - Assembly
 - Runtime Data Structures
 - The Stack

Basic x86 Concepts *(cont'd)*

- Registers:

- All processors contains small amounts of internal memory, known as registers.
- Range in size from 8 bits (1 byte) to 128 bits (16 bytes), with 32-bit registers being the most common.
- Some registers are general in purpose, while others are used for specific purposes.
- Most important register is the **EAX** register. It holds the results of arithmetic operations and function return values.

Basic x86 Concepts *(cont'd)*

- Assembly:

- Assembly is a symbolic language that is “assembled” into machine language by an assembler.
- Each processor has its own instruction set and assembly language.
- Assembly deals directly with the registers of the processor and memory locations.
- Example:

```
beagle:004031DB loc_4031DB:  
beagle:004031DB          push    3E8h          ; 1000 milliseconds  
beagle:004031E0          call   Sleep  
beagle:004031E5          jmp    short loc_4031DB
```

Basic x86 Concepts *(cont'd)*

- Runtime Data Structures:

- Object files and executables come in several formats.
- COFF (Common Object-File Format) is one such format used on Windows systems.
- These object files are separated into areas called segments.
- There are several segments that are common to all executable formats (may be named differently, depending on the compiler/linker):

`.text, .data, .rdata, .bss, heap, stack`

Basic x86 Concepts *(cont'd)*

- The Stack:
 - A LIFO (Last In First Out) structure.
 - The stack is used to store parameters, local variables, and temporary storage.
 - A new stack frame (procedure activation record) is created for each new function call.
 - The stack is accessed via 2 registers:
 - ESP (**Stack Pointer**): a runtime pointer which always points to the top of the stack.
 - EBP (**Base Pointer**): points to the base of the current stack frame and stays constant, so it's easy to refer to parameters and local variables using offsets from this pointer.

Basic x86 Concepts *(cont'd)*

- The Stack:

Virus Overview

- A **virus** is a self-replicating piece of code that attaches itself to other programs and usually requires human interaction to propagate.
- A **worm** is a self-replicating piece of code that spreads via networks and usually doesn't require human interaction to propagate.
- Most viruses today have features of both viruses and worms in one program.

Bagle Virus Disassembly

- Overview
- Analysis Resources
- Disassembly Approach
- Analysis Problems and Solutions
- Functional Flow of Bagle

Bagle Virus Disassembly *(cont'd)*

- Overview:

- The virus we chose to analyze was called **Bagle** (or Beagle), variant A.
 - It is **widespread** and **recent** (first appeared in Jan 2004).
 - It has **18 variants**, as recent as July 2004.
- It does...
 - ...use its own **SMTP engine** for propagation.
 - ...install a **backdoor**.
 - ...contact the attacker via web sites.
- The project, including gaining the necessary background knowledge, took 10 weeks to complete.

Bagle Virus Disassembly *(cont'd)*

- Analysis Resources:

- Major tools we used during analysis:

- IDA Pro: interactive disassembler and debugger.
 - dumpbin: dumps information about COFF/PE files.
 - UltraEdit: text/hex editor.
 - Virtual PC: allows you to install an OS within another OS.

- Major websites/books:

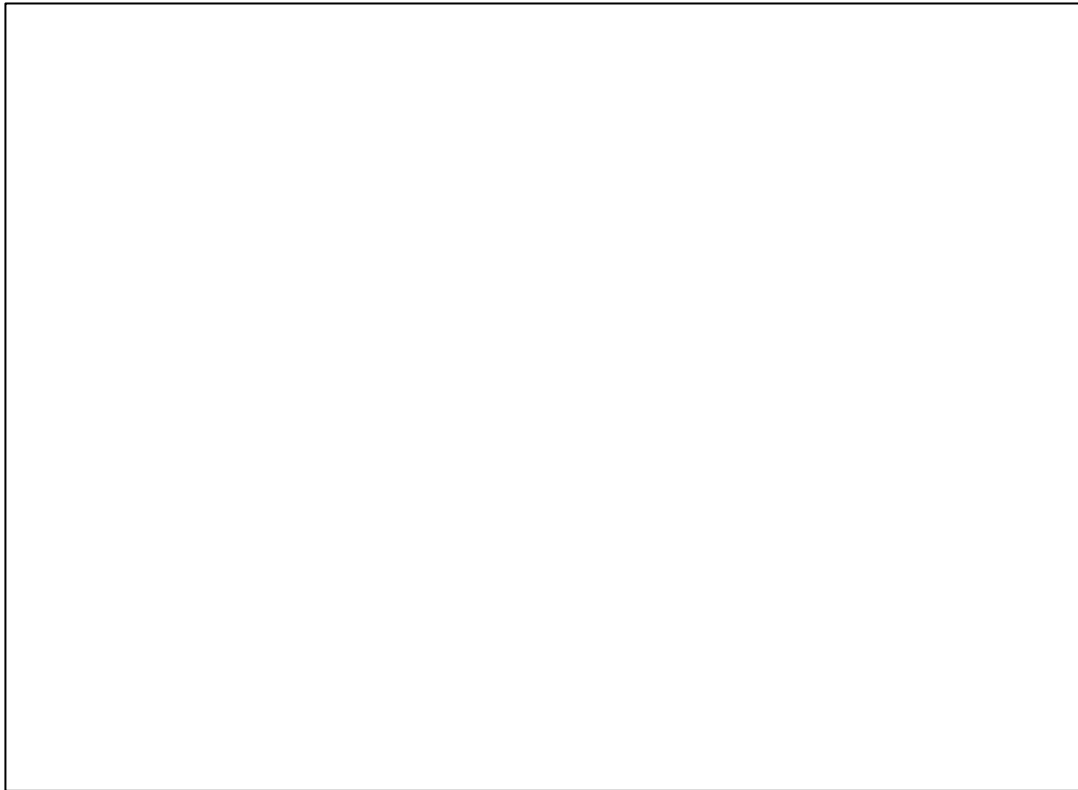
- MSDN Library: <http://msdn.microsoft.com/library/>
 - IA-32 Intel® Architecture Software Developer's Manual
Volume 2: Instruction Set Reference

Bagle Virus Disassembly *(cont'd)*

- Disassembly Approach:
 - Create a **secure environment** or host where the virus could be analyzed.
 - Setup a server to **log all virus network activity** (DNS and SMTP traffic).
 - Acquire virus.
 - Download it from <http://vx.netlux.org>
 - Wait for an infected email.
 - Search for it in Google.
 - Load the virus into IDA Pro and start analysis.
 - Use the **debugger** often.
 - Keep track of registers.

Bagle Virus Disassembly *(cont'd)*

- IDA Pro Screenshot:



Bagle Virus Disassembly *(cont'd)*

- Analysis Problems and Solutions:
 - Problem: IDA Pro didn't find main() in the virus!
 - Solution: the developer can change the start up code of his compiler and set the entry-point symbol (the function called by the start-up code) manually.
 - Problem: The virus won't run if it's after Jan 28, 2004.
 - Solution: change the date on the system or EAX register.
 - Problem: If there is no Internet connection, the virus will loop indefinitely looking for an Internet connection.
 - Solution: change the EAX register, or connect host to network.

Bagle Virus Disassembly *(cont'd)*

- Functional Flow of Bagle:
 1. First it **initializes COM** (Component Object Model) in order to allow objects to interact with each other.
 2. It checks to make sure it is before **January 28, 2004**.
 - If it's not, it won't run.
 - Or it will stop and delete itself from system.
 3. It creates a **registry entries**, copies itself to the Windows system directory, makes a base64-encoded copy of itself in memory (for the email).

Bagle Virus Disassembly *(cont'd)*

- Functional Flow of Bagle:
 4. Starts a new thread that listens on **port 6777** and accepts and processes connections.
 - Allows attacker to upload and execute programs.
 - **Allows attacker to stop & delete the virus from system.**
 5. Creates a new thread that **contacts a list of websites** every 10 minutes to inform of infection.
 - Sends port number, `uid` registry entry, and IP
 6. Creates a thread that searches fixed drives for email addresses and **emails itself** to them.
 - Uses its own SMTP engine to interact with mail servers.

Bagle Virus Disassembly *(cont'd)*

- Example of a detailed disassembly of Bagle:

ESI register points to a string of all drives on system (i.e. "A:\0C:\0D:\0")

```
beagle:00402CF5 loc_402CF5:
beagle:00402CF5     cmp     byte ptr [esi], 0      ; is the string empty?
beagle:00402CF8     jz     short loc_402D16      ; If yes, get out of loop, else continue
beagle:00402CFA     push   esi                   ; lpRootPathName
beagle:00402CFB     call   GetDriveTypeA        ; call Win32 API function
beagle:00402D00     cmp     eax, 3               ; is this a fixed drive (hard drive)?
beagle:00402D03     jnz    short loc_402D0B     ; if no, go to loc_402D0B, else
beagle:00402D05     push   esi                   ; push fixed_disk onto stack
beagle:00402D06     call   sub_402C9D           ; else call sub_402C9D(fixed_drive);
beagle:00402D0B loc_402D0B:
beagle:00402D0B     push   esi                   ; string representing non-fixed drive
beagle:00402D0C     call   lstrlenA             ; get length of that string
beagle:00402D11     add    esi, eax              ; add it to ESI register
beagle:00402D13     inc    esi                   ; move ESI to the next char (next drive)
beagle:00402D14     jmp    short loc_402CF5     ; go back and repeat
```

Conclusions

- **How do you reverse engineer a virus?**
 - This question is answered throughout the paper, specifically in Section 4 and Appendix A.
- **Can reverse engineering be done more efficiently?**
 - We estimate that Bagle could've been reverse engineered within a week, if we had previous knowledge of RCE.
 - Automation of grouping functions with parameters and “parsing” the assembly to give functional flow.
 - Automatically tracing all the branches possible.

Conclusions *(cont'd)*

- **Can reverse engineering a virus lead to better ways of detecting, preventing, and recovering from a virus and its future variants?**
 - Hidden behaviors (i.e. Bagle can be remotely removed).
 - Accesses certain websites and DNS server.
 - Probe listening ports with known commands.
 - Uses string “beagle_beagle” as the user agent for HTTP connections.
 - Uses `bbeagle.exe` as the process name.
 - Listens on port 6777 for connections.
 - Will not run if it's after Jan 28, 2004 and will delete itself if it is running after Jan 28, 2004.
 - Create a better signature (based on functional flow – FFSig)

Demo

- Demo #1:

Modify Bagle virus to make it sterile:

- Sleep for 30 seconds, not 600 seconds, when trying to contact websites.
 - *If date becomes after January 28, 2004, virus should kill and delete itself from system in 30 seconds, not 10 minutes.*
- But it still won't die in 30 seconds because other threads were running and since there is no Internet connection those threads would never release the mutex.
 - *So I modify the threads that listen on port 6777, the one that creates and sends out emails, and the one that checks what type of drive a system has. (used NOPS, JZ, JNZ, and JMPS)*
 - *The virus is now harmless, it just deletes itself after January 28, 2004.*

Demo *(cont'd)*

- Demo #2:

Connect to Bagle on port 6777 and remove it remotely:

- *Send a certain bit stream to remove virus from system.*
- *Wrote a small program to do this, called: `kill_bagle`*

More Information

- For a complete report on this project, including the full detailed technical analysis of Bagle, see my paper entitled:

Reverse Code Engineering: An In-Depth Analysis
of the Bagle Virus

- You can contact me at:
 - E-mail: krozinov@yahoo.com
 - Website: <http://rozinov.sfs.poly.edu>