## In-depth Malware Analysis

***Dawn Song***

***dawnsong@cs.berkeley.edu***



## Other Issues & Attacks against GhostBuster?

- **Malware only hides from certain processes**
  - **Solution?**
    - » **Run GhostBuster in every process**
  - **Issues?**
  - **On the other hand, such a malware is not very stealthy**
- **How well does GhostBuster approach deal with VMBR?**
- **Why do we need GhostBuster if we have ways to get real truth?**
  - **A way to pinpoint stealth behavior which is anomalous**
  - **Corollary: always be honest :-)**



## Does Getting Real Truth Solve the Problem?

- **How would you design a truly stealthy malware hiding the fact that the machine has been compromised?**
- **Getting real truth is really just the first step**
  - **Finding needle in the hay stack**

## Turning Things Around

- What would you do if you are a AV geek, and doesn't want your AV program to be killed by malware?
- Good guys have a lot to learn from bad guys
  - Let the fun begin :-)

4

## Open Mic

- Anything else you thought that's really clever in the papers?
- Anything else you didn't like about the papers?
- Any other unclear points about the papers?
- Other comments/remarks to share?

5

## In-depth Malware Analysis

- What do we want to find out about malware?
  - What inputs malware read
    - » Keystrokes
    - » Check registry key
    - » Gettimeofday
    - » Network recv
  - What outputs malware produce
    - » Write file/registry
    - » Network send
  - Relationship btw different behaviors
  - Special inputs triggering certain behaviors
  - Semantic information: DDoS? SPAM?

6

## Traditional Analysis Methods (I): Manual Analysis

- **Runs in debugger, single-step**
- **Disadvantages**
  - **Labor intensive**
  - **Can't keep up with volume of new malware samples**



## Traditional Analysis Methods (II): Static Analysis

- **Challenges**
  - **Code packing, encryption, obfuscation**
  - **What examples of obfuscation techniques can you think of?**



## Traditional Analysis Methods (III): Dynamic Testing

- **Executing in virtual machine environment**
- **Record system calls & their args**
- **Limitations**
  - **Incomplete view**
  - **Miss behaviors triggered by different environments**
    - **Certain registry key set**
    - **Certain file exists**
    - **Mutex**
    - **Network connection**
    - **Time bomb**
    - **Commands in bot programs**

## Malware Analysis Similar to Software Testing

- **Need to observe behavior under different inputs**
- **Important metric: coverage**
- **Testing:**
  - **Fuzzing: try random inputs**
    - » **Advantages:**
      - **Often easy to do**
      - **Black-box**
    - » **Disadvantages:**
      - **Difficult to cover certain paths**



## Symbolic Execution to Automatically Explore Multiple Paths

- **Idea:**
  - **Make inputs symbolic**
  - **Symbolically execute program**
  - **Build path constraints**
  - **Solve path constraints to take different branches**



## Example

```
Struct {int type; char arg[512];} cmd;
// code to set up server.
While (1) {
  read (net_sock, & cmd, sizeof(cmd));
  if (cmd.type == 0x1){
    DDoS (cmd.arg);
  } else if (cmd.type == 0x2) {
    Spam (cmd.arg);
  }  else if (cmd.type – 0x3) {
     Execute (cmd.arg);
  }  else {
     die();
  }
}
```

## Things to Take Care of

- Why path constraints?
- Efficiency to represent symbolic expressions
- What about symbolic memory addresses?
- Doing it on binary
  - DART/EXE on source code



## Challenges

- Solving constraints
  - Attacker making constraints really hard to solve
  - Examples?
- Path exploration
  - What strategies one may use to prioritize different paths?



## Open Mic

- Still lots of cool things to be done
- Other comments/remarks?

## Break Time



## Class Project (I)

- **Binary analysis**
  - **bitblaze.cs.berkeley.edu**
  - **Infrastructure to build cool stuff on top**
    - » **Well-documented**
    - » **Don't necessarily need prior experience**



## The BitBlaze Project

- **Two research focii**

1. **Design and develop the underlying BitBlaze Binary Analysis Platform**

2. **Apply the BitBlaze Binary Analysis Platform to real security problems**
   - **COTS vulnerability analysis & defense**
   - **Malicious code analysis & defense**

## BitBlaze Binary Analysis Platform

**Currently 3 components:**

1. **Vine: Static analysis component**
   - Raise assembly to Intermediate Language (IL)
   - Provides program analysis and verification routines on IL
2. **TEMU: Dynamic analysis component**
   - Whole system emulation (OS aware)
   - Dynamic analysis techniques (such as taint analysis)
3. **Rudder: Mixed execution component**
   - Mixed concrete and symbolic execution
   - Can explore code paths automatically

**Research directions:**

- How to design & combine static & dynamic analysis & other techniques (e.g., machine learning) for effective binary analysis?

19

## BitBlaze in Action (I)
## COTS Vulnerability Analysis & Protection

- **Exploit & worm defense:**
  - Worm characteristics:
    - » Exploit vulnerabilities: memory safety vulnerability
    - » Fast self-propagation, large scale
      - Slammer infected 90% of vulnerable hosts in 10 minutes, compromised hundreds of thousands of machines
  - Detect new exploits & identify root causes
  - Create signatures for vulnerabilities (IEEE S&P 2006, CSF 2007)
  - Create dynamic patches
  - Project: how to automatically create effective defense?
- **Detect deviations in protocol implementation (USENIX Security 2007, *Best Paper Award*)**
  - Create formulas representing different implementations
  - Diff formulas create candidate deviations
  - Project: scalable effective deviation detection

20

## BitBlaze in Action (II)
## Malicious Code Analysis & Defense

- **Central questions:**
  - Given a piece of (potentially malicious) code, how to determine its security-related behavior?
- **Project:**
  - BitScope, THE malicious code analysis platform
- **Example components**
  - Detect privacy-breaching malware (ACM CCS 2007)
  - Detect hidden behavior in malware
    - » Time bombs, botnets, etc.

21

## Class Project (II)

- Quantitative information flow
  - Recent first step towards quantitative information flow on binary
  - Lots of cool applications
- Explore building trusted path
  - More discussion in class later
- Explore building privacy into OS
- Binary instrumentation for OS for better understanding of OS & robustness
- Explore better attribution techniques in OS
  - More discussion in class later



## Class Project (III)

- Explore how to use multi-core for better security monitoring & forensic analysis
  - More discussion in class later
- Privacy-preserving distributed information sharing
  - How to make it practical
  - Leveraging trusted computing & secure hardware
- Authenticated data-publishing
  - Build data authentication into mash-ups



## Summary

- In-depth malware analysis
- Slides are on website
  - Need to be in berkeley domain to access it
- Next class: guest lecture on Symantec's approaches for malware analysis & defense
  - Think what questions you may want to ask speaker