NC STATE UNIVERSITY

CSC 405 Computer Security

Reverse Engineering Part 2

Alexandros Kapravelos
akaprav@ncsu.edu

Anti-Disassembly

- Against static analysis (disassembler)

- Confusion attack
  - targets linear sweep disassembler
  - insert data (or junk) between instructions and let control flow jump over this garbage
  - disassembler gets desynchronized with true instructions

Anti-Disassembly

- Advanced confusion attack
  - targets recursive traversal disassembler
  - replace direct jumps (calls) by indirect ones (branch functions)
  - force disassembler to revert to linear sweep, then use previous attack

Anti-Debugging

- Against dynamic analysis (debugger)
  - debugger presence detection techniques
    - API based
    - thread/process information
    - registry keys, process names, …
  - exception-based techniques
  - breakpoint detection
    - software breakpoints
    - hardware breakpoints
  - timing-based and latency detection

Anti-Debugging

Debugger presence checks

- Linux
  - a process can be traced only once

```
if (ptrace(PTRACE_TRACEME, 0, 1, 0) < 0)
    exit(1);
```

- Windows
  - API calls

```
OutputDebugString()
IsDebuggerPresent()
```

    ... many more ...

  - thread control block
    - read debugger present bit directly from process memory

Anti-Debugging

Exception-based techniques

SetUnhandledExceptionFilter()

After calling this function, if an exception occurs in a process that is not being debugged, and the exception makes it to the unhandled exception filter, that filter will call the exception filter function specified by the lpTopLevelExceptionFilter parameter. [source: MSDN]

– Idea

set the top-level exception filter, raise an unhandled exception, continue in the exception filter function

Anti-Debugging

Breakpoint detection

- detect software breakpoints
  - look for int 0x03 instructions

    ```
    if ((*(unsigned *)((unsigned)<addr>+3) & 0xff)==0xcc)
        exit(1);
    ```

  - checksum the code

    ```
    if (checksum(text_segment) != valid_checksum)
        exit(1);
    ```

- detect hardware breakpoints
  - use the hardware breakpoint registers for computation

Reverse Engineering

- Goals
  - focused exploration
  - deep understanding
- Case study
  - copy protection mechanism
  - program expects name and serial number
  - when serial number is incorrect, program exits
  - otherwise, we are fine
- Changes in the binary
  - can be done with hexedit or radare2

Reverse Engineering

- Focused exploration
  - bypass check routines
  - locate the point where the failed check is reported
  - find the routine that checks the password
  - find the location where the results of this routine are used
  - slightly modify the jump instruction
- Deep understanding
  - key generation
  - locate the checking routine
  - analyze the disassembly
  - run through a few different cases with the debugger
  - understand what check code does and develop code that creates appropriate keys

Malicious Code Analysis

Static analysis vs. dynamic analysis

- Static analysis
  - code is not executed
  - all possible branches can be examined (in theory)
  - quite fast
- Problems of static analysis
  - undecidable in general case, approximations necessary
  - binary code typically contains very little information
    - functions, variables, type information, …
  - disassembly difficult (particularly for Intel x86 architecture)
  - obfuscated code, packed code
  - self-modifying code

Malicious Code Analysis

- Dynamic analysis
  - code is executed
  - sees instructions that are actually executed
- Problems of dynamic analysis
  - single path (execution trace) is examined
  - analysis environment possibly not invisible
  - analysis environment possibly not comprehensive
- Possible analysis environments
  - instrument program
  - instrument operating system
  - instrument hardware

Malicious Code Analysis

- Instrument program
  - analysis operates in same address space as sample
  - manual analysis with debugger
  - Detours (Windows API hooking mechanism)

  - binary under analysis is modified
    - breakpoints are inserted
    - functions are rewritten
    - debug registers are used
  - not invisible, malware can detect analysis
  - can cause significant manual effort

Malicious Code Analysis

- Instrument operating system
  - analysis operates in OS where sample is run
  - Windows system call hooks

  - invisible to (user-mode) malware
  - can cause problems when malware runs in OS kernel
  - limited visibility of activity inside program
    - cannot set function breakpoints

- Virtual machines
  - allow to quickly restore analysis environment
  - might be detectable (x86 virtualization problems)

Malicious Code Analysis

- Instrument hardware
  - provide virtual hardware (processor) where sample can execute (sometimes including OS)
  - software emulation of executed instructions
  - analysis observes activity “from the outside”

  - completely transparent to sample (and guest OS)
  - operating system environment needs to be provided
  - limited environment could be detected
  - complete environment is comprehensive, but slower

  - Anubis uses this approach

Stealthiness

- One obvious difference between machine and emulator
  - → time of execution

- Time could be used to detect such system
  - → emulation allows to address these issues
  - → certain instructions can be dynamically modified to return innocently looking results
  - → for example, RTC (real-time clock) - RTDSC instruction

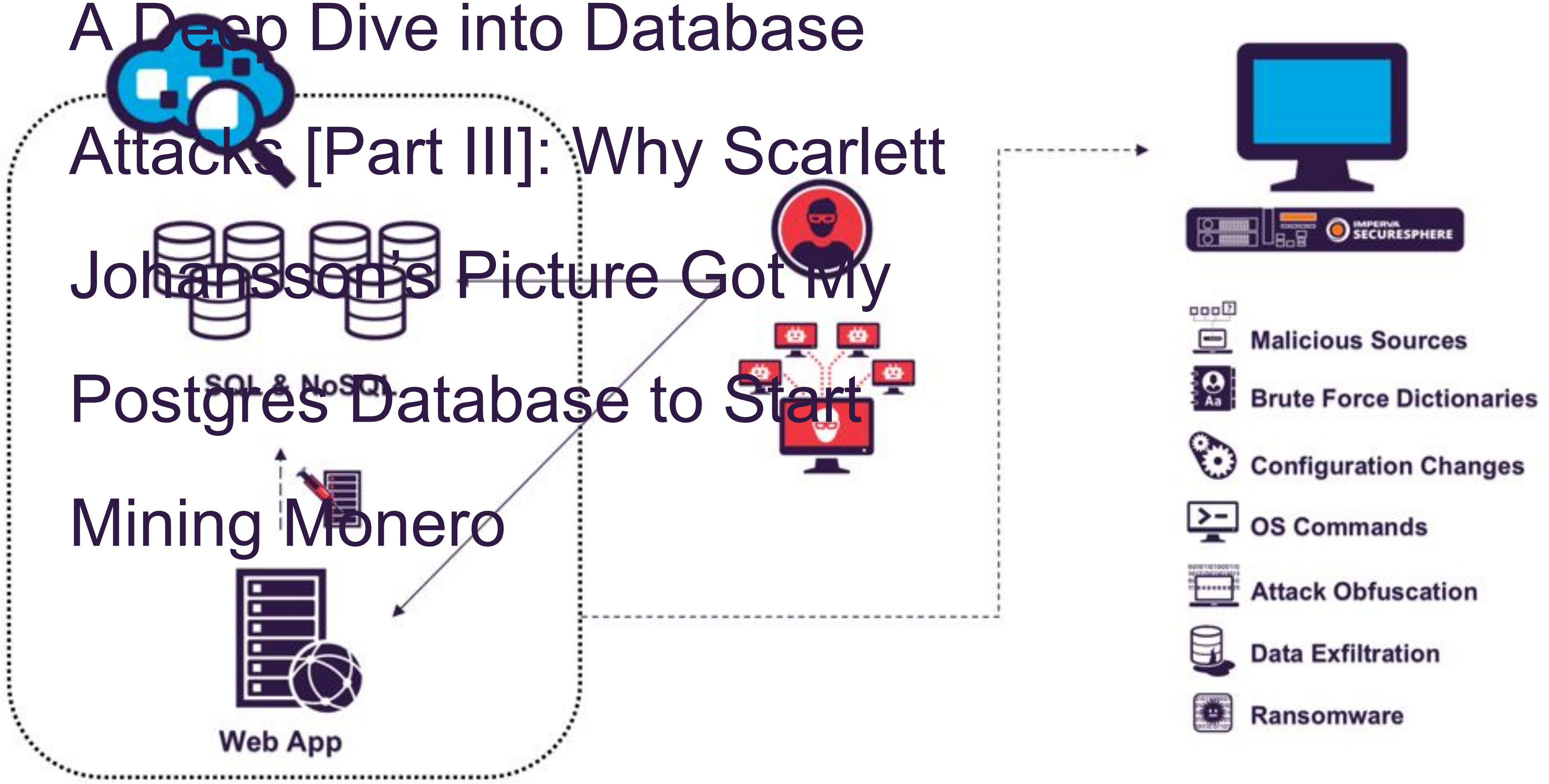
Challenges

- Reverse engineering is difficult by itself
  - a lot of data to handle
  - low level information
  - creative process, experience very valuable
  - tools can only help so much

- Additional challenges
  - compiler code optimization
  - code obfuscation
  - anti-disassembly techniques
  - anti-debugging techniques

Your Security Zen

A Deep Dive into Database Attacks [Part III]: Why Scarlett Johansson's Picture Got My Postgres Database to Start Mining Monero



source: https://www.imperva.com/blog/2018/03/deep-dive-database-attacks-scarlett-johanssons-picture-used-for-crypto-mining-on-postgre-database/