Use of Malware by Penetration Testers

Wesley McGrew
Department of Computer Science and Engineering
Distributed Analytics and Security Institute
mcgrew@cse.msstate.edu

@McGrewSecurity
Me

• Assistant Research Professor at Mississippi State University
  • CAE Cyber Operations

• Affiliations: CSE, DASI, NFTC

• Research focuses: Large-scale malware analysis, vulnerability analysis

• Freelance: Forensics, penetration testing, vulnerability analysis

• DEF CON

• @McGrewSecurity
Topic

My usual peanut butter & jelly style mashup:

Applying what I’m doing with reversing malware to penetration testing

Good Summit topic: Get smart people thinking/doing
What does your penetration testing represent?
What should your penetration testing represent?

As realistically as possible: An attack on the target organization by an advanced, funded threat.
What’s different about you vs. actual criminals…

How much of your penetration testing takes place while you’re off doing something else?
Penetration testing software (and therefore, the way you test) is *highly interactive*

Real attacks are done at-scale, and/or an eye to stealth. They are *highly automated*
Your workflow may be different, but...

- How close to “the real thing” are your C2 protocols?
- Would even a properly set up and updated AV detect you?
- …however you start, you may want to ramp up to activities that should be detected.
- …and some malware/criminal activity may be more evasive than what you normally do
- How fast do you work?
So what do we do?

• Some malware is *really* quite easy to understand and change

• Reverse engineer, patch, test, tinker
  
  • Online scanner OPSEC

• So we learn some new tricks and bring them to the table

• Not outside the realm of possibility: (class project discussion)
POS Malware Example
Acknowledgements

- Samples - @xylit0l - http://cybercrime-tracker.net
- Prior-to-now-but-post-this-work analyses
Why JackPOS?

• Current concern surrounding POS malware

• C2 availability - Ability to demonstrate a complete environment
  • From card-swipe to command-and-control

• C++ strings, STL - runtime objects make static analysis with IDA Pro a bit more awkward

• Good use case for harnesses
  • Independent memory-search functionality
Harness Design

- WinAppDbg - Python scriptable debugging

- *Really* fun library - Well-documented, lots of examples, easy to use

- Callbacks for breakpoints

```python
def patch_cnc(event):
    """
    Breakpoint: 0x00401B38
    Patch the CnC to ours
    """
    process, thread, context = get_state(event)
    original_cnc = process.peek_string(0x004339BC)
    process.write(0x004339BC, debug_cnc+'\x00')
    printModification('Modified CnC from %s to %s' % (original_cnc, debug_cnc))

    esp = context['Esp']
    process.write_dword(esp+0x04, len(debug_cnc))
    printModification('Patched length to %i' % (len(debug_cnc))

    return
```
JackPOS

- Example sample - SHA1
  9fa9364add245ce873552aced7b4a757dceceeb9e
  - Available on virusshare

- Command and Control

- PHP, Yii Framework
Command and Control

- Data model - bots, cards, commands, dumps, ranges, tracks, users
Back to the sample

- UPX (thankfully not an unpacking talk/tutorial)
  - Unpacked version crashes due the stack cookie seed address not relocating
  - Easy fix: disable ASLR (also makes our analysis easier), unset:
    - IMAGE_NT_HEADERS > IMAGE_OPTIONAL_HEADER > IMAGE_DLLCHARACTERISTICS_DYNAMIC_BASE
Setup

- String setup - c2, executable filenames, process names for memory search
- Installation (copying self)/persistence (registry)
- Harness patches -
  - Command and control
  - Installation check
  - Prevents watchdog process (and anything else from ShellExecute’ing)
Communication

• Command and Control Check-in
  • Checks C2 for http://[c2]/post_echo
    • (PostController.php responds “up”)
    • Prevents simple sandbox from getting much
  • If there’s track data, base64 it and send it
    • Harness configured to display data sent
  • Check command queue
• Hosts uniquely identify by MAC
Commands

- Credit card track theft happens without having to be commanded to do so

- Remainder of command set is simple:
  - `kill`
  - `update - (replace current install with latest from /post/download)`
  - `exec <url>`
Scraping Memory

- Get a list of functions
  - No 64-bit process
  - No processes matching internal table (system, etc)
- Iterate and search for card data using two regular-expression-esque functions
  - ISO/IEC 7813 (we can generate and instrument this)
- Harness identifies search process
- Another harness can be used to instrument the code to scan arbitrary PIDs
Level Up

- IDA Pro - Expensive, unquestionably worth it
  - Alternatives: IDA Pro Free (for learning), radare, Hopper…
  - Reference work: Eagle’s “The IDA Pro Book”
- Python - Everything scriptable with reverse engineering has a python binding
Level Up

- Learning assembly well enough to read it

  - opensecuritytraining.info

  - Intel Manuals

  - Dr. Paul Carter’s “PC Assembly Language”

- Not “Art of Assembly Language” no no no don’t
Level Up

- Actually learning reverse engineering
  - Sikorski and Honig’s “Practical Malware Analysis”
  - /r/reverseengineering
Counter-argument: Why not do this?

- It's a lot of work.
- You’d better do it right. Testing, testing, testing.
- Cleanup
Conclusions

- Raising post-exploitation game. Adding and applying some security skills that are outside of the usual penetration testing set
- Potential for making your penetration tests more realistic
- Learning something from people who put food on their table through crime
- (It’s fun.)
Discussion