LESSONS LEARNED WRITING EXPLOITS

Gerardo Richarte | gerardo.richarte@corest.com
Iván Arce | ivan.arce@corest.com

Outline

- Why write exploits?
- What to look for in an exploit?
- Lessons learned
- Conclusions

Outline

Why write exploits?

Why write exploits?

- To confirm the existence of a vulnerability
- To confirm the disappearance of a vulnerability
- To *exploit* a vulnerability
- To *consistently exploit* a vulnerability
  “Commercial Grade Exploit Code”

What to look for in an exploit?

What to look for in an exploit?

What to look for in an exploit?

- Reliability
- Low maintenance
- Fail safe
- Portability
- Short Development Cycle

Lessons Learned

Lesson 1

- Use an interpreted language
  - Short Development Cycle
  - Platform portability
  - Easy maintenance
  - Low overhead
- Choice: Python or Perl
  - Object Oriented
  - Readability

Lesson 2

- Automatic shell code generation
  - Code reuse
  - Platform independency
  - Modularization
  - Encapsulation
- Decision: build a library
  - LibEgg
  - Egg class

LibEgg

- Basic Egg
  - Win Egg
  - Unix Egg
    - Linux Egg
    - Solaris Egg
- Egg Decorators
  - XOR Egg
  - NOP Egg
  - Transport Egg
- Ancillary functions
  - setuid()
  - GetCodeAddress()

Lesson 3

- Common exploitation procedures!
  - Standardize exploit development
  - Shorten development cycle
  - Reliability?
  - Fail safe?
- Decision: structure the exploit
  - Exploit Class

Exploit Class

```
class Exploit:

    def initialSetup(self)
        return 1

    def passSetup(self)
        return 1

    def tryAttack(self)
        return 1

    def done(self)
        return(self._done)
```

Exploit Class

```
class Exploit:

    def run(self)

        self.initialSetup()
        while not self.done()
            self.passSetup()
            self.tryAttack()
```

Lesson 4

- Address brute forcing issues
  - Reliability
  - Fail safe
  - Code reuse
- Decision: DB for exploit parameters
  - Address Finder
    - Operating System, OS version , Service Packs
    - Distribution specific parameters (kernel versions, libc versions)
    - Exploit method specific parameters (GOT, destructors, libc functions, memchunks, ld.so)

Address Finder

class AddressFinder:

```
def __init__(self, OSVersion = None, ServicePack = None,
ProductVersion = None):

self.OSVersion      = OSVersion
 self.ServicePack    = ServicePack
 self.ProductVersion = ProductVersion

 self.dlls = []

def findAddressFor(self,opcode = 'CALL EBX', WantedDll
=None):
```

Address Finder

class WindowsAddressFinder(AddressFinder):

```
def __init__(self, OSVersion = None, ServicePack = None, ProductVersion = None):

    AddressFinder.__init__(self, OSVersion, ServicePack, ProductVersion)
     dll = ProgramInformation('kernel32');

    self.dlls.append(dll)
    dll.OSVersion      = '4.0';
    dll.ServicePack    = 6;
    dll.ProductVersion = '4.00';
    dll.BaseAddress = 0x77f00000
    dll.Address['JMP ESP']     = 0x77F32836
    dll.Address['JMP EDI']     = 0x77F2D148
    dll.Address['CALL EAX']    = 0x77F1A9DD
    dll.Address['JMP [EBX]']   = 0x77F0174A
    dll.Address['CALL EBX']    = 0x77F01089
    dll.Address['JMP ECX']     = 0x77F05372
    dll.Address['JMP [EAX]']   = 0x77F08070
    dll.Address['GetProcAddress'] = 0x77F1402F
    dll.Address['LoadLibraryA']   = 0x77F1382E
```

More Lessons

- Debugging
- Further exploit factorization
- Need for a common language
- Exploit Primitives?
- Exploit Transforms?

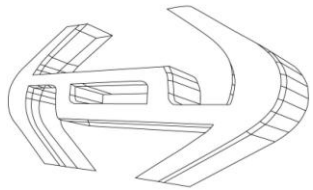
A Few Conclusions

Conclusions

- Writing “commercial grade exploit code” is quite different and a LOT more difficult than simple PoC exploits
- Once the process is engaged systematically many new ideas and techniques are discovered. Often leading to a higher level understanding of the process.
- A common framework and a common language facilitates the process.
- It is probably the most unexplored territory in the InfoSec field

Thank You!

- Iván Arce | ivan.arce@corest.com
- Gerardo Richarte | gerardo.richarte@corest.com



CORE SECURITY TECHNOLOGIES · Offices Worldwide

Headquarters
44 Wall Street | 12th Floor
New York, NY 10005 | USA
Ph: (212) 461-2345
Fax: (212) 461-2346
info.usa@corest.com

Florida 141 | 2º cuerpo | 7º piso
(C1005AAC) Buenos Aires
Tel/Fax: (54 11) 4878-CORE (2673)
info.argentina@corest.com

Rua do Rócio 288 | 7º andar
Vila Olímpia | São Paulo | SP
CEP 04552-000 | Brazil
Tel: (55 11) 3054-2535
Fax: (55 11) 3054-2534
info.brazil@corest.com

www.corest.com