Expanding the control over the operating system from the database

Bernardo Damele Assumpção Guimarães
Guido Landi

Warsaw (Poland) – November 20, 2009

Who we are

- Bernardo Damele Assumpção Guimarães
  - Proud father
  - Penetration tester / security researcher at Portcullis Computer Security Ltd
  - sqlmap lead developer
- Guido Landi
  - Reverse engineer
  - Exploit writer
  - Vulnerability researcher

Introduction

- Database management systems are powerful applications
  - Store and interact with data
  - Interact with the file system and operating system
    - When they can't by design, you can force them to
    - When they can't due to limited user's privileges, you can exploit them!

Scenario

- You have got access to a database
  - Direct access – provided account, weak passwords, brute-forcing credentials
  - SQL injection – web application, stand-alone client, cash machine ☺, …
- What to do now other than enumerating data?
  - Own the underlying operating system
  - Why not even other servers within the DMZ?

State of art – File system access

- Microsoft SQL Server
  - Read: **`BULK INSERT`**
  - Write: **`xp_cmdshell`** / **`debug.exe`**
- MySQL
  - Read: **`LOAD_FILE()`**
  - Write: **`SELECT … INTO DUMPFILE`**
- PostgreSQL
  - Read: **`COPY`** / UDF
  - Write: **`Large object's lo_export()`**

State of art – Command execution

- Microsoft SQL Server
  - **`OPENROWSET`** can be abused to escalate privileges
  - Built-in **`xp_cmdshell`** to execute commands

- Oracle
  - If you find a SQL injection in a function owned by **`SYS`** and with **`authid definer`**, you can run PL/SQL as **`SYS`**
  - Many ways to execute commands. Example:

    `DBMS_EXPORT_EXTENSION` package's **`GET_DOMAIN_INDEX_TABLES()`** function

State of art – Command execution

- MySQL and PostgreSQL support user-defined functions: custom function that can be evaluated in SQL statements
- UDF can be created from **shared libraries** that are compiled binary files
  - **Dynamic-link library** on Windows
  - **Shared object** on Linux
- PostgreSQL supports also procedural languages

Demonstration

Operating system command execution by exploiting a SQL injection vulnerability in a web application

Further information on these techniques can be found on http://tinyurl.com/sqlmap1

More than command execution

- Owning the underlying operating system is not only about command execution
- **Full-duplex connection between the attacker host and the database server**
- Database used as a stepping stone to establish this covert channel
  - Shell, Meterpreter, VNC – http://metasploit.com
  - DNS tunnel – http://heyoka.sourceforge.net

Establish the channel

- On your box
  - Forge a stand-alone payload stager with **`msfpayload`**
  - Encode it with **`msfencode`** to bypass AV
  - Run **`msfcli`** with **`multi/handler`** exploit

- On the database server
  - Upload it to the file system temporary folder
  - Execute it via UDF, **`xp_cmdshell`**, …

Getting stealth

- Anti-forensics technique as an option to upload the stand-alone payload stager executable

- On your box
  - Forge a shellcode with **`msfpayload`**
  - Encode it with **`msfencode`**
  - Run **`msfcli`** with **`multi/handler`** exploit

- On the database
  - Create a UDF that executes a payload in-memory
  - Execute the UDF providing the payload as a parameter

User-defined function `sys_bineval()`

- Execute an arbitrary payload from the database management system memory
- Features
  - Works in DEP/NX-enabled systems
  - Supports alphanumeric payloads
  - Protects the DBMS if the payload crashes
  - It does not fork a new process

`sys_bineval()` vs DEP/NX

- Use `VirtualAlloc()` to allocate an **+RWX** memory region

```
code = (char *) VirtualAlloc(NULL,
                             4096,
                             MEM_RESERVE|MEM_COMMIT,
                             PAGE_EXECUTE_READWRITE);
```

`sys_bineval()` and alphanum payloads

- Metasploit's **`msfencode`** has alphanumeric encoders to encode the payload

- Problem: It is not able to produce **pure** alphanumeric payloads due to **`get_pc()`**

`sys_bineval()` and alphanum payloads

- Solution:
  - Use the **`BufferRegister`** option

```
./msfencode BufferRegister=EAX -e x86/alpha_mixed …
```

  - Put the payload address in **`EAX`** register

```
__asm
{
        MOV EAX, [lpPayload]
        CALL EAX
}
```

sys_bineval(): avoid DBMS crash

- Spawn a new thread

```
WaitForSingleObject(CreateThread(NULL, 0,
                        ExecPayload, CodePointer,
                        0, &pID),
                        INFINITE);
```

- Wrap the payload in a **SEH** frame

```
__try {
        __asm {
                MOV EAX, [lpPayload]
                CALL EAX
        }
}
```

Demonstration

Exploit a SQL injection vulnerability in a web application to establish an out-of-band channel in-memory via a custom UDF

Hands on the Windows registry

- Microsoft SQL Server
  - Built-in stored procedures, **`xp_reg(read|write|delete)`**
- MySQL and PostgreSQL
  - Upload and execute a **`bat`** file that executes **`reg (query|add|delete)`**
  - Upload and execute a file and pass it to **`regedit`**

MS09-004: Memory corruption

- Discovered by Bernhard Mueller, it affects Microsoft SQL Server up to **2005 SP2**
- Triggered by a call to **`sp_replwritetovarbin`**
- No authentication and no privileges needed

- "*Limited Memory Overwrite Vulnerability*" could allow remote code execution
- "*Limited Memory Overwrite*"... Actually a pretty huge heap-based buffer overflow, ~4000 bytes

Exploiting MS09-004

- Target heap metadata
  - Could be hard/unreliable even if Microsoft SQL Server uses a custom allocator
- Target application specific data
  - Function pointers, C++ object pointers, etc.
- Luckily for us Microsoft SQL Server tries hard to not crash by graceful handling exceptions...

Exploiting MS09-004

- ...this gives us more than one code path to achieve code execution:
  - An almost arbitrary 4-bytes overwrite:

```
MOV DWORD PTR DS:[EAX+4], EDI
```

  - An object pointer overwrite:

```
MOV EDX,DWORD PTR DS:[ESI]
[...]
MOV EAX,DWORD PTR DS:[EDX+10]
[...]
CALL EAX
```

Bypass hardware-enforced DEP

- Use ret2libc to call **`ZwSetInformationProcess()`**
  - Make **`ESP`** point to our buffer:

```
PUSH ESI
POP ESP
RET
```

  - No need for a fake stack frame, just return in the middle of **`LdrpCheckNXCompatibility()`**

Bypass hardware-enforced DEP

- `LdrpCheckNXCompatibility()`

```
[…]
MOV DWORD PTR SS:[EBP-4],2
PUSH 4
LEA EAX,DWORD PTR SS:[EBP-4]
PUSH EAX
PUSH 22
PUSH -1
CALL ntdll.ZwSetInformationProcess
[…]
```

- DEP is now disabled for the current process
  - …then jump to the shellcode. Game over?

Avoid crash

- The original stack address is gone, **`ESP`** and **`EBP`** point to our buffer
- Even if Microsoft SQL Server tries hard to handle exceptions, it will eventually crash
- We need to restore **`ESP`** and **`EBP`**
- Is there a generic way?

Thread Environment Block

- TEB stores information about the currently running thread:

```
0:000> !teb
TEB at 7ffdd000
    ExceptionList:          0012fd04
    StackBase:              00130000
    StackLimit:             0012e000
    SubSystemTib:           00000000
    FiberData:              00001e00
    ArbitraryUserPointer:   00000000
    Self:                   7ffdd000
    EnvironmentPointer:     00000000
    ClientId:               00000d2c
    RpcHandle:              00000000
    Tls Storage:            00000000
[…]
```

TEB: Restore the Stack Pointer(s)

- Contains 3 pointers to the current thread's stack
- Addressable through the FS segment register
- Just prepend the shellcode with a little stub:

```
MOV ESP, DWORD PTR FS:[0]
MOV EBP, ESP
SUB ESP, 20
```

- Game Over!

Demonstration

Own the system by exploiting MS09-004 vulnerability via a SQL injection vulnerability in a web application with back-end Microsoft SQL Server

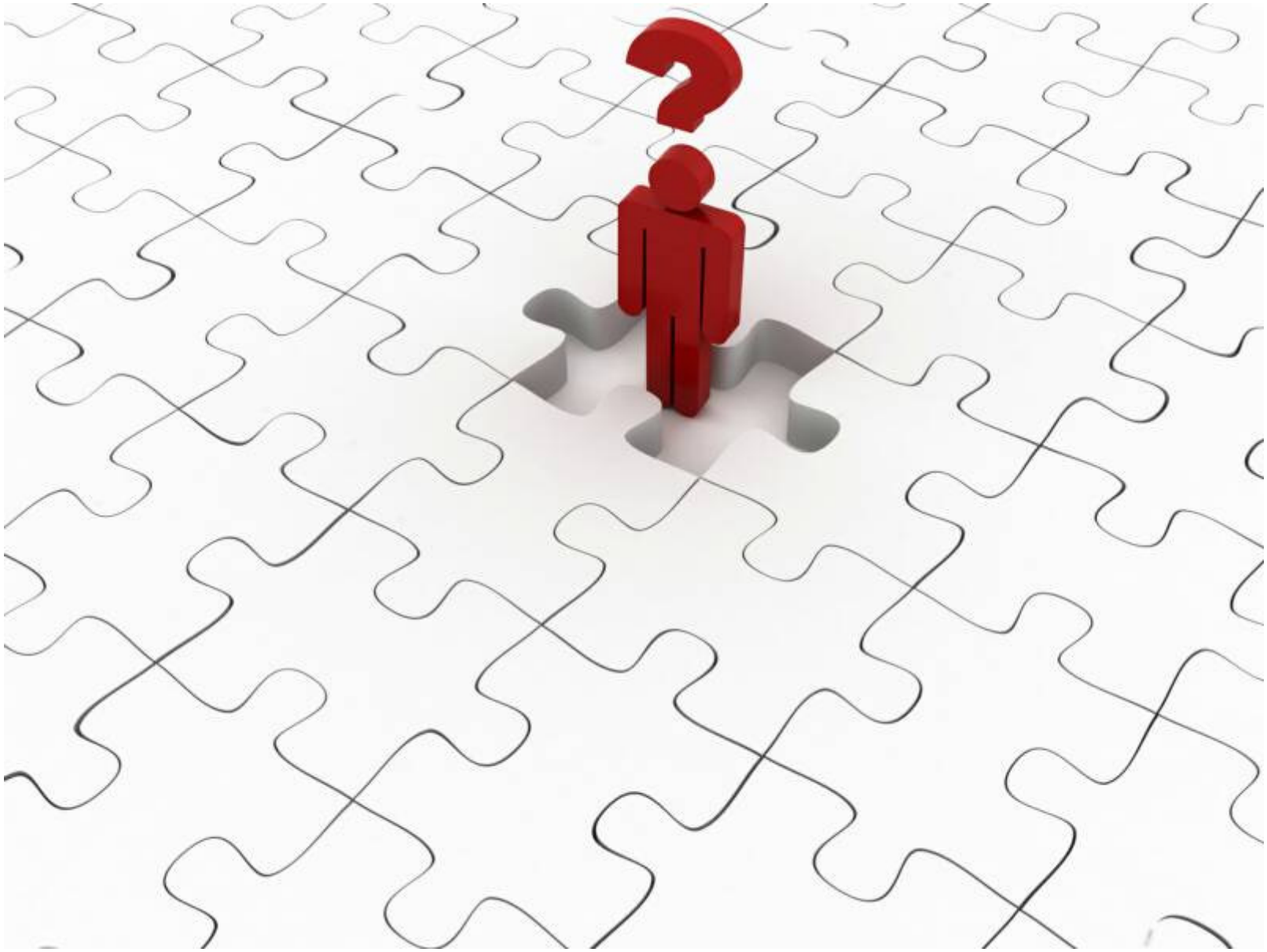
But… Wasn't it meant to deal with data?

- Once you get access to a database you can compromise the whole system in most cases
  - With a comfortable, fast and stable channel
- Once you have access to the system you can escalate privileges (token kidnapping, software bugs, kernel flaws, weak privileges, etc.)
- When you are **`root/Administrator/SYSTEM`** you can crack users' passwords or impersonate them to get access to other servers within the network perimeter

Credits

- Our mums for buying our first pre-school computers
- Alessandro Tanasi and Oliver Gruskovnjak for the technical discussions
- skape and skywing for their paper on DEP bypass
- H D Moore and the Metasploit development team
- Andrzej Targosz and the CONFidence team

Questions?



Thanks for your attention!

- Bernardo Damele Assumpção Guimarães

  bernardo.damele@gmail.com

  bda@portcullis-security.com

  http://bernardodamele.blogspot.com

  http://sqlmap.sourceforge.net

- Guido Landi

  lists@keamera.org

  http://www.pornosecurity.org

  http://milw0rm.com/author/1413