Dangling Pointer

Jonathan Afek, 2/8/07, BlackHat USA
Table of Contents

- What is a Dangling Pointer?
- Code Injection
- Object Overriding
- Demonstrations
- Remediation
- Summary
- Q&A
What is a Dangling Pointer?

Invalid Pointer:
- Dangerous
- Easy to Exploit
- Common

Diagram:
- Pointer
- Pointer
- Dangling Pointer
- Object
- Object
- New Data
What is a Dangling Pointer? - An Example

- Results:

Crash

![Internet Information Services dialog box showing an error message and options to debug, send error report, or don't send.](image-url)
What is a Dangling Pointer? - An Example

**Debugger View**

<table>
<thead>
<tr>
<th>5AA5919A</th>
<th>5D</th>
<th>POP EBP</th>
</tr>
</thead>
<tbody>
<tr>
<td>5AA5919B</td>
<td>C2 0C00</td>
<td>RETN 0C</td>
</tr>
<tr>
<td>5AA5919E</td>
<td>CC</td>
<td>INT3</td>
</tr>
<tr>
<td>5AA5919F</td>
<td>CC</td>
<td>INT3</td>
</tr>
<tr>
<td>5AA591A0</td>
<td>CC</td>
<td>INT3</td>
</tr>
<tr>
<td>5AA591A1</td>
<td>CC</td>
<td>INT3</td>
</tr>
<tr>
<td>5AA591A2</td>
<td>CC</td>
<td>INT3</td>
</tr>
<tr>
<td>5AA591A3</td>
<td>8BFF</td>
<td>MOV EDI,EDI</td>
</tr>
<tr>
<td>5AA591A5</td>
<td>56</td>
<td>PUSH ESI</td>
</tr>
<tr>
<td>5AA591A6</td>
<td>8BF1</td>
<td>MOV ESI,ECX</td>
</tr>
<tr>
<td>5AA591A8</td>
<td>8B4E 20</td>
<td>MOV ECX,DWORD PTR DS:[ESI+20]</td>
</tr>
<tr>
<td>5AA591A9</td>
<td>57</td>
<td>PUSH EDI</td>
</tr>
<tr>
<td>5AA591AC</td>
<td>6A 01</td>
<td>PUSH 1</td>
</tr>
<tr>
<td>5AA591AD</td>
<td>C706 F88A25A</td>
<td>MOV DWORD PTR DS:[ESI],w3svc.??_7WAM_RECV</td>
</tr>
<tr>
<td>5AA591AE</td>
<td>8B01</td>
<td>MOV EAX,DWORD PTR DS:[ECX]</td>
</tr>
<tr>
<td>5AA591B6</td>
<td>56</td>
<td>PUSH ESI</td>
</tr>
</tbody>
</table>

5AA591E7 | FF50 0C | CALL DWORD PTR DS:[EAX+C] |

5AA591B0 | 8B4E 20 | MOV ECX,DWORD PTR DS:[ESI+20] |

5AA591BD | E8 E496FFFF | CALL w3svc.??_DereferenceCWamInfo@@QAEXX |
| 5AA591C2 | 8BCE | MOV ECX,ESI |

5AA591C4 | E8 4BDAFFFF | CALL w3svc.??_DoCleanupOnDestroy@WAM_RECV |

5AA591C9 | 85C0 | TEST EAX,EAX |

Access violation when reading [0000000C] - use Shift+F7/F8/F9 to pass exception to program
Where are We

- What is a Dangling Pointer?
- **Code Injection**
- Object Overriding
- Demonstrations
- Remediation
- Summary
- Q&A
Code Injection - The Layout of an Object

- **Class_A:**

  ```cpp
class Class_A {
  public:
    virtual long vfunc_A1();
    virtual long vfunc_A2();
    static void sfunc_A();
  void funcA();
};
```

```
{ ...
  this.vfunc_A2();
  ...
}
```

```
... MOV EAX, [ECX]
  CALL [EAX + 4]
...
Exploit Overview:
- Free the Object
- Override the Object – covered later
- Execute a Virtual Function
Code Injection - The Double Reference Exploit

- **Injecting Code**
  - Free the Object
  - Shellcode
  - Call/Jmp ECX
  - Finding a “VFTABLE”
  - Interpreted as Code

- **Continue**
  - Automation

![Diagram showing the flow of code injection and the double reference exploit.](Diagram.png)
Code Injection - Double Inheritance

- Multiple Inheritance

```cpp
class Inherited: public Class_A, public Class_B
{
    public:
    virtual int vfunc_A2();
    virtual int vfunc_B2();
};
```
Where are We

- What is a Dangling Pointer?
- Code Injection
- **Object Overriding**
- Demonstrations
- Remediation
- Summary
- Q&A
Object Overriding

- Allocation Implementation
  - Numerous heaps
  - Two Default heaps
  - Different API
  - C-Runtime functions
    - Malloc
    - Free
    - New
    - Delete
    - Etc.
Object Overriding

- Allocation implementation details
  - Lookaside List
    - A list for each size (8-1024) and for each heap
    - First Allocation Priority
    - Merges
And Finally – Overriding

- Search for Allocations

- Static Analysis
  - Method: Disassembly
  - Restriction: Static Size
  - Validation: Controllable Content
  - Usage: Causing the Allocation

- Dynamic analysis
  - Method: API Breakpoints
  - Restriction: Static/Dynamic Size
  - Validation: Controllable Content
Object Overriding - The VFTABLE Exploit

- Exploitation:
  - Empty the Lookaside List
  - Allocate a Buffer
  - Insert Content
  - Free the Buffer

- Continue:
  - Free the Object
  - Execute a VFunc

![Diagram showing the VFTABLE exploit process]
Object Overriding - The Lookaside Exploit

- Empty the Lookaside
- Allocate Two Buffers
- Insert Shellcode
- Free One Buffer
- Free The Other
- Free The Object
- Execute the Destructor

GAME OVER!!!
Object Overriding - The Lookaside Exploit

- Executing NULL – NO Problem
Summary

- Double Reference
  - Controllable First DWORD
  - Static Address
- VFTABLE Exploit
  - Controllable Allocations
  - No First DWORD
  - Static Address
- Lookaside Exploit
  - Controllable Allocations
  - No First DWORD
  - No Static Address
  - Destructor Execution
Where are We

- What is a Dangling Pointer?
- Code Injection
- Object Overriding
- **Demonstrations**
- Remediation
- Summary
- Q&A
Demonstrations - Configuration Item

- Allocating the Object
- De-Allocation the Object
Demonstrations - Configuration Item

- Allocating User Data
Demonstrations - Configuration Item

- Executing a VFunc
Demonstrations - Configuration Item

- Putting it Together
  - De-Allocate
  - Re-Allocate
  - Execute
Demonstrations - Remote Exploit

- Another Exploit on IIS,
  but this time – a remote one
Where are We

- What is a Dangling Pointer
- Code Injection
- Object Overriding
- Demonstrations
- **Remediation?**
- Summary
- Q&A
Remediation

- Known Protection Mechanisms
  - NX Bit
  - ASLR
- VFTABLE Sanitation
- Safe Programming
Summary

- Technical Background
  - Memory Allocations
  - Objects Implementation

- Exploits
  - Double Reference Exploit
  - VFTABLE Exploit
  - Lookaside Exploit

- Demonstrations
  - Configuration Item
  - Remote Exploit

- Dangling Pointer
  - Only Object Oriented Objects
Questions

- Ask Away…