Windows New Security Features
- Control Flow Guard

FFRI, Inc
http://www.ffri.jp
About Control flow guard (Guard CF)

• **Control flow guard** made its debut at Windows 8.1 Preview release
  – It disabled on Windows 8.1 RTM (Release To Manufacturing) and Windows 8.1 releases
  – Available on Windows 10 Technical Preview and Windows 8.1 Update Pack

• We call control flow guard “**Guard CF**” in this document
  – Because acronym of control flow guard (CFG) means control flow graph generally
Notes

• Guard CF is work-in-progress feature

• We tested Windows 10 Technical Preview and Visual Studio 2015 Preview
Threat Model

- Arbitrary code execution
  - Manipulating indirect call operand

- Typical example
  - vtable overwrite
Protecting with Guard CF

- Insert check function called before indirect calls at compile time
- The check function validates indirect call target address
  - Raises violation if untrusted address’s called

```c
void myfunc1() {
    printf("myfunc1\n");
}

int main(int argc, char* argv[])
{
    void(*myfunc)();
    myfunc = myfunc1;
    (*myfunc)();
    return 0;
}
```

- Compiler inserts check code
- Linker embeds guard information

If the target is trusted function, calling (*myfunc)()

Exception has occurred if may call untrusted indirect call target address
Protecting with Guard CF (cont.)

- Guard CF trusts registered address of guard CF function table
- Guard CF function table exists PE/COFF headers which made by linker
- Windows runtime (ntdll.dll) builds trusted function bitmap from Guard CF function table at loading time

Compiler Inserts check function before indirect calls

Linker makes guard CF configuration and appends executable file headers
Guard CF in Visual Studio 2015 Preview

• Using hidden option

  cl /d2guard4 test.cpp /link /guard:cf

See also:
PE/COFF headers

- DLL Characteristics

**OPTIONAL HEADER VALUES**

10B magic # (PE32)

... 

C140 DLL characteristics
- Dynamic base
- NX compatible
- **Guard**
- Terminal Server Aware

build with guard CF option

**OPTIONAL HEADER VALUES**

10B magic # (PE32)

... 

8140 DLL characteristics
- Dynamic base
- NX compatible
- Terminal Server Aware

build without guard CF option
PE/COFF headers (cont.d)

- Load config structure in PE/COFF headers

Section contains the following load config:

0000005C size
...
0041D108 Guard CF address of check-function pointer
00000000 Reserved
0041D150 Guard CF function table
   2A Guard CF function count
00003500 Guard Flags
   CF Instrumented
   FID table present
   Protect delayload IAT
   Delayload IAT in its own section
PE/COFF headers (cont.d)

- Guard CF function table in PE/COFF headers

... Guard CF Function Table

<table>
<thead>
<tr>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>00401000</td>
</tr>
<tr>
<td>00401030</td>
</tr>
<tr>
<td>004011E0</td>
</tr>
<tr>
<td>00401270</td>
</tr>
<tr>
<td>004013F0</td>
</tr>
</tbody>
</table>

...
int main(int argc, char* argv[]) {
    void(*myfunc)();
    myfunc = myfunc1;
    (*myfunc());
    return 0;
}

Inserted Guard CF check function

Sample code

Dis-assembled view
Guard CF Function Bitmap

- Guard CF check function validates target address using bitmap
  - Bitmap is created by loader
  - Raising security assertion exceptions (int 29h) if call target not exist in bitmap

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Size</th>
<th>Committed</th>
<th>Private</th>
<th>Total WS</th>
<th>Private...</th>
<th>Share...</th>
<th>Share...</th>
<th>Lock...</th>
<th>Blocks</th>
<th>Protection</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>001F0000</td>
<td>Shareable</td>
<td>64 K</td>
<td>64 K</td>
<td>4 K</td>
<td>4 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 Read/Write</td>
<td>64-bit thread stack</td>
<td></td>
</tr>
<tr>
<td>00210000</td>
<td>Shareable</td>
<td>76 K</td>
<td>76 K</td>
<td>72 K</td>
<td>72 K</td>
<td>72 K</td>
<td></td>
<td></td>
<td></td>
<td>1 Read</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00240000</td>
<td>Thread Stack</td>
<td>256 K</td>
<td>44 K</td>
<td>44 K</td>
<td>12 K</td>
<td>12 K</td>
<td></td>
<td></td>
<td></td>
<td>3 Read/Write/Guard</td>
<td>Thread ID: 5068</td>
<td></td>
</tr>
<tr>
<td>00280000</td>
<td>Thread Stack</td>
<td>1,024 K</td>
<td>20 K</td>
<td>20 K</td>
<td>12 K</td>
<td>12 K</td>
<td></td>
<td></td>
<td></td>
<td>3 Read/Write/Guard</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00300000</td>
<td>Shareable</td>
<td>16 K</td>
<td>16 K</td>
<td>16 K</td>
<td>15 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 Read</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00390000</td>
<td>Private Data</td>
<td>8 K</td>
<td>8 K</td>
<td>8 K</td>
<td>8 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 Read/Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>003A0000</td>
<td>Mapped File</td>
<td>728 K</td>
<td>728 K</td>
<td>128 K</td>
<td>128 K</td>
<td>128 K</td>
<td></td>
<td></td>
<td></td>
<td>1 Read</td>
<td>C:\Windows\System32\locale.nis</td>
<td></td>
</tr>
<tr>
<td>00520000</td>
<td>Private Data</td>
<td>64 K</td>
<td>23 K</td>
<td>28 K</td>
<td>20 K</td>
<td>20 K</td>
<td></td>
<td></td>
<td></td>
<td>2 Read/Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00610000</td>
<td>Heap (Private Data)</td>
<td>1,024 K</td>
<td>48 K</td>
<td>48 K</td>
<td>48 K</td>
<td>48 K</td>
<td></td>
<td></td>
<td></td>
<td>2 Read/Write</td>
<td>Heap ID: 1 [COMPATABILITY]</td>
<td></td>
</tr>
<tr>
<td>00A00000</td>
<td>Image (ASLR)</td>
<td>168 K</td>
<td>168 K</td>
<td>28 K</td>
<td>112 K</td>
<td>16 K</td>
<td>96 K</td>
<td></td>
<td></td>
<td>5 Execute/Read</td>
<td>C:\Users\Yasuka\Desktop\cfgtest\bin\cfgtest.exe</td>
<td></td>
</tr>
<tr>
<td>00AE0000</td>
<td>Shareable</td>
<td>32,768 K</td>
<td>6,175 K</td>
<td>20 K</td>
<td>24 K</td>
<td>4 K</td>
<td>12 Read</td>
<td>Reserved</td>
<td>Read</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00AEE0000</td>
<td>Shareable</td>
<td>28 K</td>
<td>23 K</td>
<td>12 K</td>
<td>12 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00AF5000</td>
<td>Shareable</td>
<td>84 K</td>
<td></td>
<td>8 K</td>
<td>8 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B0A000</td>
<td>Shareable</td>
<td>8 K</td>
<td></td>
<td>8 K</td>
<td>8 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B9C000</td>
<td>Shareable</td>
<td>24,300 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C:\Windows\System32\locale.nis</td>
<td></td>
</tr>
<tr>
<td>022C7000</td>
<td>Shareable</td>
<td>5,580 K</td>
<td>5,580 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0231A000</td>
<td>Shareable</td>
<td>24 K</td>
<td>24 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C:\Windows\System32\locale.nis</td>
<td></td>
</tr>
<tr>
<td>02340000</td>
<td>Shareable</td>
<td>348 K</td>
<td>348 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02377000</td>
<td>Shareable</td>
<td>16 K</td>
<td>16 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>023B8000</td>
<td>Shareable</td>
<td>128 K</td>
<td>128 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>023BB000</td>
<td>Shareable</td>
<td>44 K</td>
<td>44 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>023C6000</td>
<td>Shareable</td>
<td>2,152 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75B0C000</td>
<td>Image (ASLR)</td>
<td>1,404 K</td>
<td>1,404 K</td>
<td>26 K</td>
<td>238 K</td>
<td>18 K</td>
<td>220 K</td>
<td>220 K</td>
<td>4 Execute/Read</td>
<td>C:\Windows\SysWOW64\KernelBase.dll</td>
<td></td>
<td></td>
</tr>
<tr>
<td>76DC3000</td>
<td>Image (ASLR)</td>
<td>896 K</td>
<td>579 K</td>
<td>16 K</td>
<td>156 K</td>
<td>12 K</td>
<td>144 K</td>
<td>144 K</td>
<td>12 Execute/Read</td>
<td>C:\Windows\SysWOW64\kernel32.dll</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Limitation

• CF Guard protects indirect call only
  – Indirect jump and return is not protected

• Code reuse attack mitigation is limitedly
  – Guarded functions could be called by any indirect caller
Ref: Control flow integrity (CFI)

- Control flow integrity (CFI) restricts indirect branch (jmp, call, ret) source and destination
  - Microsoft researcher published this research in 2005

- CFI implementation uses binary translation and static control flow analysis

```c
bool lt(int x, int y) {
    return x < y;
}
bool gt(int x, int y) {
    return x > y;
}
sort2(int a[], int b[], int len){
    sort( a, len, lt );
    sort( b, len, gt );
}
```
Relation between Guard CF and CFI

- CFI guaranteed stronger control flow integrity than Guard CF
- But, CFI needs binary translation and many function insertions
  - It has an impact on performance and binary compatibility
- Guard CF simplified CFI that checks trustworthiness of call target

```c
bool lt(int x, int y) {
    return x < y;
}
bool gt(int x, int y) {
    return x > y;
}
sort2(int a[], int b[], int len){
    sort( a, len, lt );
    sort( b, len, gt );
}
sort2():
    ...
    call sort
    ...
    call sort
    ...
    ret..
sort():
    ...
    call check_func
call [ebp+var_4]
    ...
    ret
    ...
    ret
lt():
    ...
    ...
    ret
gt():
    ...
    ...
    ret
```

Inserts check function at compile time
Conclusion

• Introducing Control flow guard (Guard CF) design and implementation
  – To enable Guard CF for existing source code, application developers re-compile program using compiler option and linker option with Guard CF aware compiler

• Microsoft attempting to put Guard CF into practical use
  – It based on control flow integrity research over a decade
References

  (2014/12/19 viewed)


• Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti, “Control-Flow Integrity”,
  ACM CCS’05, November 2005
Contact Information

E-Mail : research—feedback@ffri.jp
Twitter : @FFRI_Research