Reversing Windows patches for n-days

Mitchell Adair
csg.utdallas.edu
9/19/2013
Outline

- About Me
- Background
- Tools
- Case Study
  - MS13-071
  - Getting started
  - Differing
  - Hitting the code
  - Where to now?
- Conclusion
About me
About me

- Founded CSG
- Binaries ftw
  - CTFs, wargames, vr, etc.
  - All great practice
- Currently doing security research
  - Break things, on the beach ;)
  - CSG + all these activities do pay off
Background
Given an unpatched binary, and a patched one...

We can find the differences between them to uncover patched vulnerabilities

Just like diffing a text file

# ‘diff file1 file2’
Background

- Second Tuesday of every month
- Commonly reversed and sold with exploit packs
- Good ones are still valuable
- Turn around time much quicker than looking for 0day.

Background

- Two release types
  - GDR, Security updated only
  - QFE, Security + functionality
  - To minimize differences, GDR is preferable (when available).
- XP patches come in an easy to extract .exe format
- XP patches (sometimes) contain less changes, and may be easier to diff
Tools

- Reversing a binary is hard
- Reversing two binaries is harder
- No need to do this though...
  - Using the right tools for this job can help narrow down small portions of the binary to focus on
- BinDiff by Zynamics, stand alone or IDA plugin, great, but not free
- PatchDiff2, IDA plugin, good, free
- DarunGrim, good, free
Most of these provide the same base features, though BinDiff and DarunGrim provide some helpful statistics and heuristics.

Common features (terminology varies):
- Identical functions, same in both binaries
- Matched functions, same function is in both binaries, but changes have been made
- Unmatched functions, unique to one binary or the other
Tools

- Some unique features to BinDiff
  - Similarity and confidence scores
  - Show functions with structural and/or instruction changes
    - Structural, new basic blocks, etc.
    - Instructions, change ‘push eax’ to ‘push esi’, or ‘jz’ or ‘jnz’, etc.

- Some unique features to DarunGrim
  - Manage large repositories of binaries (update your repo every patch Tuesday, and easily diff any versions you have stored)
  - Attempts to sort functions of interest heuristically
    - If you’re wondering where to start, and don’t have BinDiff, start here
Tools - Function comparison in BinDiff

<table>
<thead>
<tr>
<th>Similarity</th>
<th>Confidence</th>
<th>Address</th>
<th>Primary Name</th>
<th>Address</th>
<th>Primary Type</th>
<th>Secondary Name</th>
<th>Type</th>
<th>Basic Blocks</th>
<th>Jumps</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.12</td>
<td>0.27</td>
<td>5B63D9C6</td>
<td>?GetDispNames@CObjectManager@W@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>?GetDispNames@CThemeManager@W@</td>
<td>Normal</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0.12</td>
<td>0.15</td>
<td>5B64E6F3</td>
<td>?Release@CObjectManager@W@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>?Release@CObjectManager@W@</td>
<td>Normal</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0.13</td>
<td>0.15</td>
<td>5B650132</td>
<td>?Release@CAccessibleWrapper@AUGKYZ</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>?Release@CAccessibleWrapper@AUGKYZ</td>
<td>Normal</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0.13</td>
<td>0.27</td>
<td>5B6536E4</td>
<td>?Invoke@CObjectManager@W@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>?Invoke@CObjectManager@W@</td>
<td>Normal</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0.20</td>
<td>0.33</td>
<td>5B6648B0</td>
<td>_GetIntFromHsString @</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>_GetIntFromHsString @</td>
<td>Normal</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0.30</td>
<td>0.92</td>
<td>5B63C3C0</td>
<td>__imp__SyncObjW@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>__imp__SyncObjW@</td>
<td>Normal</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>0.31</td>
<td>0.92</td>
<td>5B63C21C</td>
<td>__imp__SyncObjW@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>__imp__SyncObjW@</td>
<td>Normal</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0.73</td>
<td>0.73</td>
<td>5B63D356</td>
<td>?GetDispNames@CThemeManager@W@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>?GetDispNames@CThemeManager@W@</td>
<td>Normal</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0.73</td>
<td>0.73</td>
<td>5B63D37A</td>
<td>?Invoke@CObjectManager@W@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>?Invoke@CObjectManager@W@</td>
<td>Normal</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0.92</td>
<td>0.92</td>
<td>5B63C220</td>
<td>__imp__SyncObjW@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>__imp__SyncObjW@</td>
<td>Normal</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0.73</td>
<td>0.73</td>
<td>5B6648B0</td>
<td>_GetIntFromHsString @</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>_GetIntFromHsString @</td>
<td>Normal</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0.37</td>
<td>0.73</td>
<td>5B6659C2</td>
<td>?ApplyWindow@CThemeManager@W@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>?ApplyWindow@CThemeManager@W@</td>
<td>Normal</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0.50</td>
<td>0.73</td>
<td>5B64C041</td>
<td>_LoadWallpaperAsynC@CProcessThermometer @</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>_LoadWallpaperAsynC@CProcessThermometer @</td>
<td>Normal</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0.38</td>
<td>0.73</td>
<td>5B648660</td>
<td>_InitGeneralDialog@CGeneralDialog@QOEVP @</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>_InitGeneralDialog@CGeneralDialog@QOEVP @</td>
<td>Normal</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0.38</td>
<td>0.73</td>
<td>5B6535E4</td>
<td>__imp__SyncObjW@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>__imp__SyncObjW@</td>
<td>Normal</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0.38</td>
<td>0.73</td>
<td>5B6535E4</td>
<td>__imp__SyncObjW@</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>__imp__SyncObjW@</td>
<td>Normal</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0.63</td>
<td>0.73</td>
<td>5B6568F9</td>
<td>_GetPlus91ThemesDir@QYPAGCH @</td>
<td>5B63D9C6</td>
<td>Normal</td>
<td>_GetPlus91ThemesDir@QYPAGCH @</td>
<td>Normal</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

Note: The screenshot shows a comparison table with columns for Similarity, Confidence, Address, Primary Name, Address, Primary Type, Secondary Name, Type, Basic Blocks, and Jumps.
Background

- Okay, I want some n-days, now what?
  - Overall list
  - Will take you to https://support.microsoft.com/kb/2864063
    - The specific bug you are interested in
  - These guys do a quick summary of each bug, sometimes useful information
  - Webcast + slide deck going over every bug, very useful information (sometimes)
  - Usually the same information, but every now and then they let something slip ;}
Getting Started
Getting Started

- Some useful information to collect
  - What operating system(s) does this affect
  - What files are changed
    - Which two files do I actually diff?!
  - What is the attack surface (if provided)
  - What is the type of bug (if provided)
  - Are there already public proof of concepts?
## Bulletin Information

### Executive Summaries

The following table summarizes the security bulletins for this month in order of severity.

For details on affected software, see the next section, **Affected Software**.

<table>
<thead>
<tr>
<th>Bulletin ID</th>
<th>Bulletin Title and Executive Summary</th>
<th>Maximum Severity Rating and Vulnerability Impact</th>
<th>Restart Requirement</th>
<th>Affected Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS13-069</td>
<td>Cumulative Security Update for Internet Explorer (2870699)</td>
<td>Critical Remote Code Execution</td>
<td>Requires restart</td>
<td>Microsoft Windows, Internet Explorer</td>
</tr>
</tbody>
</table>

---

**Getting Started - Pick one**
## Getting Started - OS & previous file

### What OS / Software is affected?

The following table has been tested to determine which versions or editions are affected. Other versions or editions are either past their support life cycle or are not affected. To determine the support life cycle for your software version or edition, see Microsoft Support Lifecycle.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Maximum Security Impact</th>
<th>Aggregate Severity Rating</th>
<th>Updates Replaced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows XP Service Pack 3 (2854063)</td>
<td>Remote Code Execution</td>
<td>Important</td>
<td>None</td>
</tr>
<tr>
<td>Windows XP Professional x64 Edition Service Pack 2 (2854063)</td>
<td>Remote Code Execution</td>
<td>Important</td>
<td>None</td>
</tr>
<tr>
<td>Windows Server 2003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Server 2003 Service Pack 2</td>
<td>Remote Code Execution</td>
<td>Important</td>
<td>None</td>
</tr>
<tr>
<td>Windows Server 2003 x64 Edition Service Pack 2 (2854063)</td>
<td>Remote Code Execution</td>
<td>Important</td>
<td>None</td>
</tr>
<tr>
<td>Windows Server 2003 with SP2 for Itanium-based Systems (2854063)</td>
<td>Remote Code Execution</td>
<td>Important</td>
<td>None</td>
</tr>
<tr>
<td>Windows Vista</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Vista Service Pack 2 (2854063)</td>
<td>None</td>
<td>No severity rating[^1]</td>
<td>None</td>
</tr>
<tr>
<td>Windows Vista SP1 Edition Service Pack 2</td>
<td>None</td>
<td>No severity rating[^1]</td>
<td>None</td>
</tr>
</tbody>
</table>

[^1]: Severity not applicable.
From the /kb/ page, specific files that are changed are listed.
Getting Started
Watch the webcast and checkout the slide deck... seriously
Free information.
Getting Started

The SRD blog sometimes a little more information than the security bulletin.

Might drop a few key pieces of information.
Getting Started

- We’ll pick MS13-071
  - I picked something local and small, hoping to show some useful results for this presentation
- OS: Windows XP, Server 2003, ...
- File(s): themeui.dll
- Type: Vulnerability type (heap, format string, etc.) unknown, but it’s related to the parsing of .theme files
- No public PoCs
Getting Started

- Download the patch
- For XP, this is easy
- Click the OS link

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Maximum Security Impact</th>
<th>Aggregate Severity Rating</th>
<th>Updates Replaced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows XP Service Pack 3 (2854063)</td>
<td>Remote Code Execution</td>
<td>Important</td>
<td>None</td>
</tr>
<tr>
<td>Windows XP Professional x64 Edition Service Pack 2 (2854063)</td>
<td>Remote Code Execution</td>
<td>Important</td>
<td>None</td>
</tr>
<tr>
<td>Windows Server 2003</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Downloaded the patch

Extracted

Note, we only have QFE version
Getting Started

- There is only the QFE version, no biggie
  - There will be more differences since some functionality is changed
- themeui.dll is the only file provided, as expected
Diffing
Wow... 12 years old. Since no previous patch was provided, I’m just using what’s on my box.
Diffing

- The PatchDiff2 plugin is ran from inside one binary opened with IDA, and pointed to the saved .idb file to be compared against.
- BinDiff is similar, or can take 2 .idb files in the stand alone version
- DarunGrim is special...
Out of about 1,250 functions, there are only about 150 that are changed, and another roughly 90 removed/added.

Removes about 90% of the binary right away.

There’s still no way I’m going through 150+ functions though, this is where BinDiff’s or DarunGrim’s heuristics and statistics can help you chose where to start.

---

PatchDiff Plugin v2.0.10
Copyright (c) 2010-2011, Nicolas Pouvesle
Copyright (c) 2007-2009, Tenable Network Security, Inc
---

Scanning for functions ...
parsing second idb...
parsing first idb...
diffing...
Identical functions: 1015
Matched Functions: 156
Unmatched Functions 1: 54
Unmatched Functions 2: 38
done?
So, let’s just say you pick a function that looks interesting.
Diffing

- Before patch

```c
int __stdcall CThemeManager::GetIDsOfNames(int a1, int a2, int a3, int a4, int a5, int a6)
{
    return CImpIDelegate::GetIDsOfNames(a1 - 36, a2, a3, a4, a5, a6);
}
```

- After patch

```c
signed int __stdcall CThemeManager::GetIDsOfNames(int a1, int a2, LPCWSTR *a3, int a4, int a5, int a6)
{
    signed int result; // eax@3
    if ( !a4 || StrCmpIW(*a3, L"GetSelectedSchemeProperty") )
        result = 0x80070005;
    else
        result = CImpIDelegate::GetIDsOfNames(a1 - 28, a2, a3, a4, a5, a6);
    return result;
}
```
Hmm, they’re checking that arg4 is not null, and arg3 points to a specific string

Verifying the function arguments before passing them along

And that return value, 0x80070005 is E_ACCESSDENIED

How would we hit that code path?
Diffing - Code paths

- IDA has a new view called Proximity Browser
  - View -> Open Subviews -> Proximity Browser
- You can show/hide parents/children to nodes
- You can add specific functions to the view, etc.
- IDA Toolbag (google it) also has functionality to do the same
- You could also press ‘x’ a bunch of times
Diffing - Code paths
Diffing - Code paths
Diffing - Code paths
Hmm, so where do those arguments end up going. The patch seems to care about their contents, where are they later used in that function?

They end up being passed to a vtable call

- Time for C++ reversing, or debugging
- Since there are several functions to go through, this goes onto the back burner

```c
if ( v10 >= 0 )
{
  v10 = (__stdcall **)(int, int, int, int)((DWORD *)v7 + 4)(v7, a3, a4, a5);
  (*(void (__stdcall **))(int))((DWORD *)v7 + 8)(v7);
}
return v10;
```
Diffing - Round 2

- **Before patch**

```c
int __stdcall CThemeManager::Invoke(int a1, int a2, int a3, int a4, int a5, int a6, int a7, int a8, int a9)
{
    return CImpIDispatch::Invoke(a1 - 36, a2, a3, a4, a5, a6, a7, a8, a9);
}
```

- **After patch**

```c
signed int __stdcall CThemeManager::Invoke(int a1, int a2, int a3, int a4, int a5, int a6, int a7, int a8, int a9)
{
    signed int result; // eax82

    if ( a2 == 150 )
        result = CImpIDispatch::Invoke(a1 - 28, 150, (const void **)&a3, a4, a5, a6, a7, a8, a9);
    else
        result = 0x80070005;
    return result;
}
```
Diffing - Round 2

- More argument verification
- Returning access denied again if arguments don’t meet some requirement
- Could be interesting, and a few minutes of following the arguments to the next function are worth it
- But again, there are several functions being changed here (not always the case), so let’s keep trying
**StrCpyN function**

Copies a specified number of characters from the beginning of one string to another.

**Note**  Do not use this function or the StrNCpy macro. See Remarks for alternative functions.

**Remarks**

**Security Warning:** Using this function incorrectly can compromise the security of your application. The copied string is not guaranteed to be null-terminated. Consider using one of the following alternatives: StringCbCopy, StringCbCopyEx, StringCbCopyN, StringCbCopyNEx, StringCchCopy, StringCchCopyEx, StringCchCopyN, StringCchCopyNEx. You should review Security Considerations: Microsoft Windows Shell before continuing.
int __stdcall CPreviewTheme::LoadWallpaperAsync(int a1, LPCWSTR psz2, int a3)
{
    void *pContext; // esi@1 MAPDST

    // 0x40 = LPTR; Combines LMEM_FIXED and LMEM_ZEROINIT. (Allocates fixed memory. Initializes memory contents to zero.)
    pContext = LocalAlloc(0x40u, 532u);
    if ( pContext )
    {
        *(DWORD *)((DWORD *)((DWORD *)((DWORD *)((DWORD *)(pContext + 2)) + 32)) = a1;
        *(DWORD *)((DWORD *)((DWORD *)((DWORD *)((DWORD *)(pContext + 2)) + 32)) = a3;
        StrCpyNW((LPWSTR)pContext + 6, psz2, 260); // <-- Not guaranteed to be null terminated

        // second arg here is pContext
        if ( !SHQueueUserWorkItem(UpdateWallProc, pContext, 0, 0, 0, 0, 0) )
        {
            LocalFree(pContext);
        }
    return 0;
}
int __stdcall CPreviewTheme::LoadWallpaperAsync(int a1, const unsigned __int16 *a2, int a3, int use_html)
{
    HLOCAL pContext; // edi@1
    int queue_return; // eax@4

    // 0x40 = LPTR: Combines LMEM_FIXED and LMEM_ZEROINIT. (Allocates fixed memory. Initializes memory contents to zero.)
    pContext = LocalAlloc(0x40u, 532u);
    if (pContext)
    {
        *((DWORD *)pContext) = *((DWORD *)(a1 + 32));
        *((DWORD *)(pContext + 2)) = a3;

        // NEW®, safe copy with error checks instead of StrCpyNW
        if ( (StringCopyWorkerW((unsigned __int16 *)pContext + 6, 260u, a2) & 0x80000000) == 0 )
        {
            // API change, arg4 determines HTML or not
            if ( use_html )
                queue_return = SHQueueUserWorkItem(UpdateWallProcHTML, pContext, 0, 0, 0, 0, 0);
            else
                queue_return = SHQueueUserWorkItem(UpdateWallProc, pContext, 0, 0, 0, 0, 0);
            if ( !queue_return )
                LocalFree(pContext);
        }
        return 0;
    }
}
unsigned int __stdcall StringCopyWorkerW(unsigned __int16 *dest, unsigned int size, const unsigned __int16 *src)
{
    unsigned int result; // eax01

    result = 0;
    if (size)
    {
        do
        {
            if (!src)
            {
                break;
                *dest = *src;
                ++dest;
                ++src;
                --size;
            }
            while (size);
            if (!size)
            {
                --dest;
                result = 0x8007007A;
                // <-- "The data area passed to a system call is too small."
            }
            *dest = 0;
        } else
        {
            result = 0x80070057;
        }
    return result;
}
Diffing - Round 3

- pContext = LocalAlloc(0x40u, 532u);
- ...
- StrCpyNW((LPWSTR)pContext + 6, psz2, 260);
  - Don’t let HexRays confuse you, this is starting 12 bytes into pContext
  - LPWSTR is a wide character, 16 bytes

```assembly
mov  eax, [esp+8+arg_0]
mov  eax, [eax+20h]
mov  [esi], eax
mov  eax, [esp+8+arg_8]
push 164h          ; cchMax
push [esp+0Ch+psz2] ; psz2
mov  [esi+8], eax
lea  eax, [esi+0Ch]
push eax            ; psz1
call ds:_imp__StrCpyNW@12 ; StrCpyNW(x,x,x)
```
Diffing - Round 3

- So it’s
  - StrCpyNW((LPWSTR)pContext + 6, psz2, 260);
  - 532 - 12 = 520
    - Size remaining in pContext
  - 260 * 2 = 520
    - Number of bytes allowed to copy
    - The ‘W’ in this function means wide, aka 2 bytes
    - 260 wide characters, 520 bytes
- So, the change in call doesn’t look to be checking for a buffer overflow (in this case, the new helper function definitely does though)
- But, it is checking to make sure the string is null terminated
Who cares about null termination?

- Any following string operations on this buffer could be wrong
  - Depending on what is directly after this buffer
- strlen, strcat, str*...
  - Will all go until a null byte is reached, 2 null bytes in the case of a wide string
  - It could be a while before 2 more null bytes are found directly after this string
- You might be able to get your data right after this string on the heap!
- Also, memory leaks
  - These are super valuable
Hmm, did they replace any other unsafe string operations?

- Let’s find xrefs on `StringCopyWorkerW`
- Wow
- Over 80 direct replacements of string copy operations on wide characters
- … this is a gold mine.
AddBackslash is a new function, that safely appends to the end of a function

Previously, size is **not** checked, and a ‘\’ is blindly appended
Diffing - Round 4

- This function has at least 10 string function calls replaced with new, safe functions.
- That screenshot is only the top third of the function...
- Two local stack buffers are having a lot of string operations performed on them, and they’re back to back on the stack

```
<table>
<thead>
<tr>
<th>esp</th>
<th>Locals above</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>String2</td>
</tr>
<tr>
<td></td>
<td>String1</td>
</tr>
<tr>
<td></td>
<td>Locals below</td>
</tr>
</tbody>
</table>
```

No guarantee of null termination + Lots of string copies / concats = Bugs

Please overwrite me!
How do we hit this code?
Hitting the code
- .theme files are executed by double clicking, or just running directly in the shell
Hitting the code

- In ProcessExplorer (System Internals Suite) you can see the full command line that gets executed.
- And it has themeui.dll loaded.
- Looks promising.
Hitting the code

- From windbg, we open rundll32.exe (from the previous slide’s command line) and pass it all the arguments listed
- Once inside, we break on the themeui.dll library loading
  - `sxe ld:themeui.dll`
- From here, let’s see which exported functions from the dll are hit
- You can find this list in IDA
- Exports subview

```
0:000> bl
0 e 5ba789d7 0001 (0001) 0:**** themeui!DllInstall
1 e 5ba789e1 0001 (0001) 0:**** themeui!DllGetClassObject
2 e 5ba778e9 0001 (0001) 0:**** themeui!DllUnregisterServer
3 e 5ba78b0d 0001 (0001) 0:**** themeui!DllMain
4 e 5ba78abf 0001 (0001) 0:**** themeui!DllAddRef
5 e 5ba78ac8 0001 (0001) 0:**** themeui!DllRelease
6 e 5ba78b71 0001 (0001) 0:**** themeui!DllCanUnloadNow
7 e 5ba7732 0001 (0001) 0:**** themeui!DllRegisterServer
```

```
Breakpoint 3 hit
eax=00000000 ebx=5ba60000 ecx=5ba90008 edx=00000000 esi=00000001 edi=0000000
```

```
Breakpoint 1 hit
eax=000a510 ebx=00000000 ecx=000a5170 edx=00000000 esi=0007e014 edi=000a5170
eip=5ba78ee1 esp=0007f74 ebp=0007f8c iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 cr0=000000246
themeui!DllGetClassObject:
5ba78ee1 8b ff  mov eax,edi edi,edi
```
Where to now?
Where to now?

- There are some *very* interesting functions doing some *very* unsafe string operations.
- We can figure out the call graph to hit those functions statically
- We can debug the program and try to hit functions of interest
- We can statically reverse some overflows or string vulnerabilities
- We can fuzz the hell out of .theme files
  - Has no one ever done this or something?
  - I’m betting it takes about 30 seconds to get a crash lol
- And I only looked at the first dozen or so xrefs to the new safe copy function, there could be even more juicy functions doing horrible things...)
Conclusion
Conclusion

- After a short period of time we’ve identified obvious vulnerabilities, and other potential spots of interest as well
- We can actively hit the dll we’re targeting
- It’s probably safe to say there is a heap or stack overflow, depending where the particular function(s) allocate their buffers from
- And, potentially, a memory leak (again, very helpful)
- Sometimes this is much, much harder. Sometimes it’s not. Your target, bug type, etc. will all have an influence over this.
- Hop in IRC (irc.oftc.net, #utdcsg) and feel free to ask questions
- Be sure to give me a shout out in the PoC (better hurry) ;)

```c

```