DISTRIBUTING THE RECONSTRUCTION OF HIGH-LEVEL INTERMEDIATE REPRESENTATION FOR LARGE SCALE MALWARE ANALYSIS

Alexander Matrosov (@matrosov)
Eugene Rodionov (@vxradius) ¹
Gabriel Negreira Barbosa (@gabrielnb)
Rodrigo Rubira Branco (@BSDaemon)

{alexander.matrosov || gabriel.negreira.barbosa || rodrigo.branco}
*noSPAM* intel.com
¹ rodionov *noSPAM* eset.com
We don’t speak for our employer. All the opinions and information here are of our responsibility (actually no one ever saw this talk before).

So, mistakes and bad jokes are all OUR responsibilities
Previous years...
We learned! (Pictures from the back intentionally omitted) ...
Or Not! (Thanks to the smoke and fire detection mechanism)
Agenda

- Introduction / Motivation
- Objectives
- Methodology
- Distributing IDA Pro (with Decompiler)
- Results
- Validating the Methodology and Toolset
- Resources
- Conclusions
- Acknowledgments
Introduction / Motivation

- Number of new malware samples grows at an absurd pace

- We still see words such as ‘many’ instead of the actual number of analyzed samples

- Assumptions without concrete data supporting them

- INDUSTRY-RELATED RESEARCH NEEDS RESULTS, THUS NOT PROMISING POINTS ARE NOT LOOKED AFTER
Objectives

- **Demonstrate** the possibility of in-depth large-scale malware analysis

- **Distribute and scale** IDA Pro (with Decompiler) to leverage its functionalities for automated malware analysis

- **Share with the community** the obtained results:
  - IDA Pro IDBs, plugins and scripts
  - Intermediate representation
  - MS Visual C++ reconstructed types
  - And more...
Methodology: Highlights

- Analyzed 32-bit and x86-64-bit PE not-packed samples from public sources
- No malware size limitations at all
- Preference on MS Visual C++ samples because of HexRaysCodeXplorer OO types reconstruction feature
- Details on the infrastructure already discussed in Black Hat Las Vegas 2012 presentation
Methodology: Overview of the process

Phase 1: Collect samples
- Pre-process samples and collect millions of 32-bit and x86-64-bit not-packed PE malware samples

Phase 2: Extract information
- Run different malware analysis algorithms on the collected samples and store results on the filesystem.

Phase 3: Analyze and parse information
- Parse and structure the results.

Phase 4: Generate statistics and charts
- Generate statistics and charts based on structured information.
Methodology: Only static analysis

- We only used static analysis
- Not detectable by malware... unless it exploits the analysis environment!
- Prone to anti-disassembly tricks
- Has some limitations... but powerful tools and techniques are available
- IDA Pro rocks!! 😊
Methodology: Malware analysis algorithms

- HexRaysCodeXplorer (by @REhints) used for:
  - Ctrees* for some IDA-recognized functions
  - MS Visual C++ object-oriented types REconstruction

- Ctrees depth analysis
  - Highly-modified version of pathfinder by @devttyS0

- AES-NI and GETSEC detection

- OO “this” usage study

- Crypto usage detection based on IdaScope by @push_pnx

* - ctrees is the intermediate representation in Hex-Rays decompiler
Constraints and Limitations: Dumping Ctrees

- Iterate through recognized routines in idb
- Process first 60 routines of size larger than 0x160 bytes
- Process first 30 crypto (using AES-NI) routines
- Process first 40 other functions bigger than 0x60 bytes

Enumerate routines

Obtain IR
- Decompile routine to get ctree (IR)
- Serialize ctree to string

Ctree normalization
- See implementation of `ctree_dumper_t::filter_citem()`
- Use normalized ctree for comparison
Constraints and Limitations:
VTBL reconstruction algorithm

Detect VTBL
- Find all calls with “this” pointer to an offset within “.rdata”/”.data” and data sections
- Find all xrefs to virtual tables

Recognize layout
- Calculate size of virtual tables
- Recognize all virtual methods

Add new VTBL Type
- Create new structure for VTBL layout representation
Constraints and Limitations: Complex types REconstruction algorithm

Detect Type
- Find pointers to possible type instances
- Find initialization routine entry point

Recognize Type layout
- Find all references to possible type address space
- Find all xrefs to the attributes of the identified type
- Reconstruct data flow for the identified type

Add new Type definition
- Create new local type if it has more than 3 attributes
Constraints and Limitations: Ctrees Depth Analysis

Enumerate code xrefs to the routine

- Use breadth-first search algorithm
- Limit: 100 nodes

Get statistics

- Distance from entry point
- Depth counter
- Number of xrefs
Constraints and Limitations: AES-NI and GETSEC Detection

- **Analyze code sections**
  - Entry point section is always analyzed

- **Scan first 512 Kb of sections**
  - Disassemble with linear sweep
  - Reject if disassembly > 20 Mb

- **Detect instructions**
  - Check for GETSEC and all AES-NI instructions
  - Reject match if a “bad” is present in 15 surrounding instructions
Constraints and Limitations: C++ “this” usage study

- Check up to 5000 call instructions
- Scan 5 instructions preceding the call
- Check ECX loads ("mov" and "lea")
- Compute percentage of calls "loading" ecx
Distributing IDA Pro: Highlights

- **Unexpected performance benefits on IDA because the information is structured**
  - But we also came across some disadvantages: SDK is complex, function signatures change from version to version and is not fully documented

- **Good performance in commodity hardware**

- **C-based plugins are usually not compatible with Linux/Mac**
  - Portability efforts are required
Distributing IDA Pro: Highlights

- IDA plugins are usually not made to scale
  - Target single-sample analysis
  - Focus on users interacting with IDA Pro interface
- Automated malware analysis exercises much more the internal plugin flows than manual analysis
  - As a result, corner cases and bugs were identified in many plugins including HexRaysCodeXplorer
Results
Pre-processing – Total: 7,829,441

Packed x Not-packed

Not-packed: 31%
Packed: 69%

Not-packed MS Visual C++ prevalence

Not-packed: 13%
Visual C++: 87%

Other compilers: 22%
AES-NI Usage (IDA Pro x standalone)

IDA Pro x Linear sweep approach

- IDA Pro AES-NI identification
- Linear sweep approach
GETSEC Usage

- Using GETSEC: 4%
- Not using GETSEC: 96%
### C++ “this” Usage Study – Top 10 Percentages

<table>
<thead>
<tr>
<th>% calls loading ECX</th>
<th>Prevalence (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7.420991619</td>
</tr>
<tr>
<td>18</td>
<td>5.845574961</td>
</tr>
<tr>
<td>30</td>
<td>5.810101164</td>
</tr>
<tr>
<td>10</td>
<td>5.247588099</td>
</tr>
<tr>
<td>16</td>
<td>4.788962581</td>
</tr>
<tr>
<td>5</td>
<td>4.468431488</td>
</tr>
<tr>
<td>3</td>
<td>4.348707424</td>
</tr>
<tr>
<td>19</td>
<td>3.988901769</td>
</tr>
<tr>
<td>20</td>
<td>3.905284962</td>
</tr>
<tr>
<td>46</td>
<td>3.193908642</td>
</tr>
</tbody>
</table>
Ctrees: Top 10 repeated ctrees (with repetition number). Total: 8,422,576

<table>
<thead>
<tr>
<th>Number of repetitions</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>40606</td>
<td>0.482109036</td>
</tr>
<tr>
<td>38800</td>
<td>0.460666665</td>
</tr>
<tr>
<td>34718</td>
<td>0.412201683</td>
</tr>
<tr>
<td>20190</td>
<td>0.239712886</td>
</tr>
<tr>
<td>19999</td>
<td>0.237445171</td>
</tr>
<tr>
<td>17635</td>
<td>0.209377749</td>
</tr>
<tr>
<td>17060</td>
<td>0.202550859</td>
</tr>
<tr>
<td>14959</td>
<td>0.177605996</td>
</tr>
<tr>
<td>14439</td>
<td>0.171432113</td>
</tr>
<tr>
<td>14072</td>
<td>0.167074776</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>232478</strong></td>
</tr>
<tr>
<td></td>
<td><strong>2.760176934</strong></td>
</tr>
</tbody>
</table>
Unique Ctrees: Repeated x Not-Repeated

Ctrees repetition

30% Ctrees with no repetitions
70% Ctrees with repetitions
Ctrees: samples with repeated x non-repeated ctrees

- Malware with repeating ctrees: 9%
- Malware with no ctrees repetition: 91%
Ctrees reaching EP + avg + std of their depth

EP reached → Average depth: 5.1940 (standard deviation: 2.3588)
82,646 or 0.98% of ctrees are directly under the EP
## Ctrees max parents (code xref) – Top 10

<table>
<thead>
<tr>
<th>Number of parents</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>11126</td>
<td>1</td>
</tr>
<tr>
<td>10989</td>
<td>3</td>
</tr>
<tr>
<td>9463</td>
<td>1</td>
</tr>
<tr>
<td>9023</td>
<td>1</td>
</tr>
<tr>
<td>8907</td>
<td>1</td>
</tr>
<tr>
<td>8837</td>
<td>2</td>
</tr>
<tr>
<td>8794</td>
<td>1</td>
</tr>
<tr>
<td>8226</td>
<td>1</td>
</tr>
<tr>
<td>7536</td>
<td>1</td>
</tr>
<tr>
<td>6917</td>
<td>5</td>
</tr>
</tbody>
</table>
VALIDATING THE METHODOLOGY AND TOOLSET

ANALYSIS OF C++ TARGETED MALWARE
Modern C++ Malware in Targeted Attacks

- Stuxnet relations
- Duqu relations
- Equation relations
- Animal Farm family
Animal Farm Case Study
Animal Farm* Case Study

- Discovered by CSEC as operation SNOWGLOBE
- Samples: NBOT, Dino, Babar, Bunny, Casper
- Written in MS Visual C++

- CSEC assesses, with moderate certainty, SNOWGLOBE to be a state-sponsored CNO effort, put forth by a French intelligence agency

Casper vs. Dino in HexRaysCodeXplorer

Casper’s virtual function tables:

Dino’s virtual function tables:
Casper vs. Dino: RUNKEY

Defines how the dropper interacts with the Windows Registry:

✓ API – call Windows Registry APIs directly
✓ BAT – modify Windows registry in a batch file using “reg” commands
✓ REG – modify Windows registry by using “reg” command in a command prompt
✓ WMI – modify Windows registry by using StdRegProv class

```c
struct IRunkeyHandler
{
    LPVOID addKey;
    LPVOID deleteKey;
    LPVOID queryKey;
    LPVOID destructor;
};
```

Casper vs. Dino: AUTODEL

Defines how dropper removes itself from machine after its execution

✓ DEL – remove itself by using command prompt
✓ API – remove itself by calling MoveFileEx
✓ WMI – remove itself by using command prompt created through create method of the Win32_Process WMI class

```c
struct IAutoDelHandler {
    LPVOID delete;
    LPVOID deleteAscii;
    LPVOID destructor;
};
```
Object Instantiation: Constructors

Casper’s RUNKEY constructor:

```c
FN_BuildAVStrat();
if ( AV_STRATEGY_RUNKEY_API )
{
    v2 = operator new(0xCu);
    v3 = v2;
    if ( v2 )
    {
        FN_GetRunKey(v2);
        *v3 = &TABLE_RUNKEY_API;
        LABEL_18:
        *v1 = v3;
        return v1;
    }
    goto LABEL_17;
}
if ( AV_STRATEGY_RUNKEY_REG )
{
    v4 = operator new(0xCu);
    v3 = v4;
    if ( v4 )
    {
        FN_GetRunKey(v4);
        *v3 = &TABLE_RUNKEY_REG;
        goto LABEL_18;
    }
    goto LABEL_16;
}
LABEL_17:
    v3 = 0;
    goto LABEL_18;
}
```

Dino’s RUNKEY constructor:

```c
sub_1005F80(&a1->vftbl_1_0106A0D0);
a1->vftbl_1_0106A0D0 = &TABLE_RUNKEY_API;
v16 = 0;
sub_1005F80(&a1->vftbl_2_0106AAE0);
a1->vftbl_2_0106AAE0 = &TABLE_RUNKEY_REG;
sub_1005F80(&a1->vftbl_3_0106A9F0);
a1->vftbl_3_0106A9F0 = &TABLE_RUNKEY_BAT;
sub_1005F80(&a1->vftbl_4_0106AB00);
a1->vftbl_4_0106AB00 = &TABLE_RUNKEY_DEFAULT;
LOBYTE(v16) = 3;
v1 = FN_GetAPIModules();
v11 = *(v1);
v12 = *(v1 + 4);
v13 = *(v1 + 8);
v2 = 0;
v14 = *(v1 + 12);
v3 = *(v1 + 16);
v7 = &a1->vftbl_4_0106AB00;
v8 = &a1->vftbl_1_0106A0D0;
v9 = &a1->vftbl_2_0106AAE0;
v10 = &a1->vftbl_3_0106A9F0;
v15 = v3;
if ( v13 )
{
    v11 = *v1;
v12 = *(v1 + 4);
v13 = *(v1 + 8);
v4 = *(v1 + 12);
v15 = *(v1 + 16);
v14 = v4;
do
    v5 = v13 >>> v2++;
    while ( !(v5 & 1) && v2 < 8u );
}
result = a1;
a1->field_0 = (&v7)[v2];
```
Object Instantiation: Type REconstruction

Casper’s RUNKEY constructor:

```c
FN_BuildAVStrat();
if (AU_STRATEGY_RUNKEY_API)
{
    v2 = operator new(0xCu);
}
```

Dino’s RUNKEY constructor:

```c
struct struct_name_RUNKEY
{
    struct_name_RUNKEY_UTABLE_0_004175FC *vftbl_0_004175FC;
    int field0;
    int field1;
};
```

```c
FN_GetRunKey(v4);
*v3 = &UTABLE_RUNKEY_REG;
goto LABEL_18;
```

```c
v13 = *(v1 + 8);
v4 = *(v1 + 12);
v15 = *(v1 + 16);
v14 = v4;

v5 = v13 >> v2++;
while (!((v5 & 1) && v2 < 8u))
{
    result = a1;
    a1->field_0 = ( &v7 )[v2];
}
```
Dino vs. NBOT in HexRaysCodeXplorer

Dino’s virtual function tables:

- 0x106aad8 - 0x106aad0: UTABLE_RUNKEY_INTERFACE methods count: 4
- 0x106aa0 - 0x106aa0: UTABLE_RUNKEY_API methods count: 4
- 0x106aa0 - 0x106aa0: UTABLE_RUNKEY_REG methods count: 4
- 0x106aa0 - 0x106aa0: UTABLE_RUNKEY_BAT methods count: 4
- 0x106ab0 - 0x106ab10: off_106ab0 methods count: 4
- 0x106ab10 - 0x106ab2c: UTABLE_SERVICE_INTERFACE methods count: 7
- 0x106ab2c - 0x106ab48: UTABLE_SERVICE_API methods count: 7
- 0x106ab48 - 0x106ab64: UTABLE_SERVICE_SC methods count: 7
- 0x106ab64 - 0x106ab80: off_106ab64 methods count: 7
- 0x106ab80 - 0x106ab90: UTABLE_AUTODEL_INTERFACE methods count: 2
- 0x106ab90 - 0x106ab90: UTABLE_AUTODEL_API methods count: 2
- 0x106ab90 - 0x106ab90: UTABLE_AUTODEL_DELAY methods count: 2
- 0x106ab90 - 0x106ab90: UTABLE_AUTODEL_DELAY methods count: 2
- 0x106ab90 - 0x106ab90: off_106ab98 methods count: 2
- 0x106ab98 - 0x106aac4: UTABLE_AOO methods count: 8

NBOT’s virtual function tables:

- 0x43d95c - 0x43d964: const wmiException::‘vtable’ methods count: 2
- 0x43d980 - 0x43d988: const NBOT_Handler::‘vtable’ methods count: 2
- 0x43d990 - 0x43d998: const CTFC_AbstractSocket::‘vtable’ methods count: 15
- 0x43d9cc - 0x43da00: const CTFC_Socket::‘vtable’ methods count: 15
- 0x43da0c - 0x43da18: const CTFC_HTTP_Request::‘vtable’ methods count: 3
- 0x43da1c - 0x43da30: const CTFC_HTTP_Form::‘vtable’ methods count: 5
- 0x43da34 - 0x43da48: const CTFC_HTTP_Form::‘vtable’ methods count: 5
- 0x43da4c - 0x43da60: const CTFC_HTTP_Form_Multipart::‘vtable’ methods count: 5
- 0x43da64 - 0x43da80: const NBOT_COM::‘vtable’ methods count: 2
- 0x43da70 - 0x43da78: const NBOT_ATCLEAR::‘vtable’ methods count: 2
- 0x43da7c - 0x43da84: const NBOT_AT::‘vtable’ methods count: 2
- 0x43da80 - 0x43da90: const NBOT_PING::‘vtable’ methods count: 2
- 0x43da94 - 0x43da9c: const NBOT_EXEC::‘vtable’ methods count: 2
- 0x43daa0 - 0x43daa8: const NBOT_HTTP_FLOOD::‘vtable’ methods count: 2
- 0x43daac - 0x43db04: const NBOT_TCP_FLOOD::‘vtable’ methods count: 2
- 0x43dbac - 0x43dcd4: const NBOT_TCP_FLOOD::‘vtable’ methods count: 3
- 0x43daca - 0x43daa0: const NBOT_WEB_FLOOD::‘vtable’ methods count: 3
- 0x43da00 - 0x43daa0: const NBOT_WEB_FLOOD_FLOOD::‘vtable’ methods count: 3
- 0x43daa0 - 0x43daa0: const NBOT_WEB_POST_FLOOD::‘vtable’ methods count: 3
- 0x43daa0 - 0x43daa0: const NBOT_WEB_PostFLOOD::‘vtable’ methods count: 3
- 0x43daa0 - 0x43daa0: const NBOT_STATISTICS::‘vtable’ methods count: 2
- 0x43daa0 - 0x43daa0: const NBOT_KILLER::‘vtable’ methods count: 2
- 0x43daa0 - 0x43daa0: const NBOT_MUTATE::‘vtable’ methods count: 2
- 0x43daa0 - 0x43daa0: const NBOT_UPLOADER::‘vtable’ methods count: 2
- 0x43daa0 - 0x43daa0: const NBOT_UPDATE::‘vtable’ methods count: 2
### Exploring NBOT’s RTTI

<table>
<thead>
<tr>
<th>VTable</th>
<th>Methods</th>
<th>Flags</th>
<th>Type</th>
<th>Hierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0043D98C</td>
<td></td>
<td></td>
<td>CTFC_AbstractSocket</td>
<td>CTFC_AbstractSocket:</td>
</tr>
<tr>
<td>0043D96B</td>
<td>1</td>
<td></td>
<td>CTFC_Anti_AV_Mailslot</td>
<td>CTFC_Anti_AV_Mailslot: CTFC_Anti_AV_Interface;</td>
</tr>
<tr>
<td>0043D970</td>
<td>1</td>
<td></td>
<td>CTFC_Anti_AV.NULL</td>
<td>CTFC_Anti_AV_NULL: CTFC_Anti_AV_Interface;</td>
</tr>
<tr>
<td>0043DA34</td>
<td>5</td>
<td></td>
<td>CTFC_HTTP_Form</td>
<td>CTFC_HTTP_Form: CTFC_HTTP_Forms, CTFC_HTTP_Request;</td>
</tr>
<tr>
<td>0043DA4C</td>
<td>5</td>
<td></td>
<td>CTFC_HTTP_Form_Multipart</td>
<td>CTFC_HTTP_Form_Multipart: CTFC_HTTP_Forms, CTFC_HTTP_Request;</td>
</tr>
<tr>
<td>0043DA1C</td>
<td>5</td>
<td></td>
<td>CTFC_HTTP_Forms</td>
<td>CTFC_HTTP_Forms: CTFC_HTTP_Request;</td>
</tr>
<tr>
<td>0043DA0C</td>
<td>3</td>
<td></td>
<td>CTFC_HTTP_Request</td>
<td>CTFC_HTTP_Request:</td>
</tr>
<tr>
<td>0043D9CC</td>
<td>M</td>
<td></td>
<td>CTFC_StandardSocket</td>
<td>CTFC_StandardSocket: CTFC_AbstractSocket, CNoImport;</td>
</tr>
<tr>
<td>0043DAAC</td>
<td>2</td>
<td></td>
<td>NBOT_ASP_FLOOD</td>
<td>NBOT_ASP_FLOOD: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DA7C</td>
<td>2</td>
<td></td>
<td>NBOT_AT</td>
<td>NBOT_AT: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DA70</td>
<td>2</td>
<td></td>
<td>NBOT_ATCLEAR</td>
<td>NBOT_ATCLEAR: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DAB4</td>
<td>2</td>
<td></td>
<td>NBOT_COM</td>
<td>NBOT_COM: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DB00</td>
<td>2</td>
<td></td>
<td>NBOT_CONFIG</td>
<td>NBOT_CONFIG: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DA94</td>
<td>2</td>
<td></td>
<td>NBOT_EXEC</td>
<td>NBOT_EXEC: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DAA0</td>
<td>2</td>
<td></td>
<td>NBOT_HTTP_FLOOD</td>
<td>NBOT_HTTP_FLOOD: NBOT_Handler;</td>
</tr>
<tr>
<td>0043D980</td>
<td>2</td>
<td></td>
<td>NBOT_Handler</td>
<td>NBOT_Handler:</td>
</tr>
<tr>
<td>0043DAF4</td>
<td>2</td>
<td></td>
<td>NBOT_KILLER</td>
<td>NBOT_KILLER: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DAB8</td>
<td>2</td>
<td></td>
<td>NBOT_PING</td>
<td>NBOT_PING: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DAE8</td>
<td>2</td>
<td></td>
<td>NBOT_STATISTICS</td>
<td>NBOT_STATISTICS: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DAB8</td>
<td>3</td>
<td></td>
<td>NBOT_TCP_FLOOD</td>
<td>NBOT_TCP_FLOOD: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DB18</td>
<td>2</td>
<td></td>
<td>NBOT_UPDATE</td>
<td>NBOT_UPDATE: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DB0C</td>
<td>2</td>
<td></td>
<td>NBOT_UPLOAD</td>
<td>NBOT_UPLOAD: NBOT_Handler;</td>
</tr>
<tr>
<td>0043DAC8</td>
<td>3</td>
<td></td>
<td>NBOT_WEB_FLOOD</td>
<td>NBOT_WEB_FLOOD: NBOT_TCP_FLOOD, NBOT_Handler;</td>
</tr>
<tr>
<td>0043DAD8</td>
<td>3</td>
<td></td>
<td>NBOT_WEB_POST_FLOOD</td>
<td>NBOT_WEB_POST_FLOOD: NBOT_TCP_FLOOD, NBOT_Handler;</td>
</tr>
</tbody>
</table>
Type REconstruction: CTFC_HTTP_Form_Multipart

Dino

```c
struct_name_2 = __thiscall FN_BuildCTFC_HTTP_Form_Multipart(int *this, char *a2)
{
    int v2; // edi@1
    unsigned int v3; // ST10 @05
    unsigned __int16 u4; // ax@05
    int v5; // eax@05
    char u7; // [sp+18h] [bp-110h]@05
    int u8; // [sp+12h] [bp-4h]@01

    v2 = this;
    sub_10183700(this, this);
    v8 = 0;
    *(DWORD *)(v2 - 01) = -CreateMutexV(0, 0, 0);
    *(DWORD *)(v2 + 200) = 0;
    *(DWORD *)(v2 + 28h) = 0;
    *(DWORD *)(v2 + 300) = 0;
    *(DWORD *)(v2 + 296) = 0;
    *(DWORD *)(v2 + 292) = malloc(0x1000u);
    LOBYTE(u8) = 1;
    if ( a2 )
    {
        if ( *(DWORD *)(v2 + 8) )
        {
            free(*(void **)(v2 + 8));
            *(DWORD *)(v2 + 8) = 0;
        }
        *(DWORD *)(v2 + 8) = _strdup(a2);
    }
    *(WORD *)(v2 + 4) = 1;
    v3 = (unsigned int)(unsigned __int16)GetTickCount() >> 2;
    v4 = GetTickCount();
    FN_print_1("multipart/form-data; boundary="", v2 + 24);
    v5 = *(DWORD *)(v2 + 12);
    sub_10090000("Content-Type", &v7);
    *(DWORD *)(v2 + 16) = 0;
    *(DWORD *)(v2 + 20) = 0;
    return (struct_name_2 *)v2;
}
```

NBOT

```c
struct_name_2 = __thiscall FN_BuildCTFC_HTTP_Form_Multipart(int *this, char *a2)
{
    int v2; // edi@1
    __int16 u3; // ax@05
    unsigned __int16 u4; // ax@05
    int v5; // ecx@05
    int u6; // eax@05
    char u7; // [sp+18h] [bp-110h]@05
    int u8; // [sp+12h] [bp-4h]@01

    v2 = this;
    sub_10183700(this, this);
    v9 = 0;
    *(DWORD *)(v2 - 01) = CreateMutexV(0, 0, 0);
    *(DWORD *)(v2 + 200) = 0;
    *(DWORD *)(v2 + 28h) = 0;
    *(DWORD *)(v2 + 300) = 0;
    *(DWORD *)(v2 + 296) = 0;
    *(DWORD *)(v2 + 292) = malloc(0x1000u);
    LOBYTE(v9) = 1;
    if ( a2 )
    {
        if ( *(DWORD *)(v2 + 8) )
        {
            free(*(void **)(v2 + 8));
            *(DWORD *)(v2 + 8) = 0;
        }
        *(DWORD *)(v2 + 8) = _strdup(a2);
    }
    *(WORD *)(v2 + 4) = 1;
    v3 = GetTickCount();
    v4 = GetTickCount();
    sub_145000(v4, v2 + 24);
    sub_145000(v5, &u8);
    v6 = *(DWORD *)(v2 + 12);
    sub_145000("Content-Type", &u8);
    *(DWORD *)(v2 + 16) = 0;
    *(DWORD *)(v2 + 20) = 0;
    return (struct_name_2 *)v2;
}
```
Type REconstruction:
CTFC_HTTP_Form_Multipart

Dino

NBOT

```c
struct struct_name_2 {__thiscall FM_BuildCTFC_HTTP_Form_Multipart(int this, char *a2) {
  int u2; // edb1
  unsigned int u3; // ST10 A95
  unsigned __int16 u4; // ax05
  int u5; // eax05
  char u7; // [sp+18h] [bp-110h]+35
  int u8; // [esp+25h] [bp-44h]+35
}
```

```c
struct struct_name_2 {__thiscall FM_BuildCTFC_HTTP_Form_Multipart(int this, char *a2) {
  int u2; // edb1
  __int16 u3; // ax05
  unsigned __int16 u4; // ax05
  int u5; // edx05
  int u6; // eax05
  char u8; // [sp+18h] [bp-110h]+35
```

```
sub_1009E0("Content-Type", &u7);
*(DWORD *)((u2 + 16) - 8); *(DWORD *)((u2 + 20) - 8);
return (struct_name_2 *)&u2;
```
<table>
<thead>
<tr>
<th>Function</th>
<th>NBOT</th>
<th>Casper</th>
<th>Bunny</th>
<th>Babar</th>
<th>Dino</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmiException</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>basic_AvWmiManager</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>basic_WmiManager</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTFC_HTTP_Form</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTFC_HTTP_Forms</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTFC_HTTP_Form_Multipart</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTFC_HTTP_Request</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTFC_AbstractSocket</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTFC_StandardSocket</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunKeyApi</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunKeyBat</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunKeyReg</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunKeyWmi</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunKeyDefault</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AutoDelApi</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AutoDelDel</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AutoDelWmi</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AutoDelDefault</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NBOT</td>
<td>Casper</td>
<td>Bunny</td>
<td>Babar</td>
<td>Dino</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>----------------</td>
<td>-------</td>
<td>---------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>NBOT</td>
<td></td>
<td>6 shared</td>
<td>3 shared</td>
<td>3 shared</td>
<td>6 shared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>custom types</td>
<td>custom types</td>
<td>custom types</td>
<td>custom types</td>
</tr>
<tr>
<td>Casper</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15 shared</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>custom types</td>
</tr>
<tr>
<td>Bunny</td>
<td></td>
<td></td>
<td></td>
<td>3 shared</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>custom types</td>
<td></td>
</tr>
<tr>
<td>Babar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dino</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We demonstrated that IDA Pro scale really well and all its powerful features can be used in automated malware analysis systems.

CALL TO ACTION: IDA Pro plugin developers to start adding batch mode switches and optimize the algorithms.

Want to run your IDA plugin on millions of malwares? Let us know! 😊
Resources

Presentation, code and instructions on how to download samples, IDBs and outputs will be available at:

https://github.com/REhints/blackhat2015
Finally plugin support Linux/Mac/Windows

Options for analysis in IDA batch mode

Multiple bug fixes and code review

Improvements for Types and VTBL’s reconstruction

New Features:

- dump Ctrees information for additional analysis
- dump all reconstructed types information

https://github.com/REhints/HexraysCodeXplorerer
Acknowledgements

Personally to Ilfak Guilfanov (@ilfak) and Hex-Rays team for supporting this research

All the researchers releasing malware-related techniques!!!
The new RE book is coming soon!

Rootkits and Bootkits
Reversing Modern Malware and Next Generation Threats
Alex Matrosov, Eugene Rodionov, and Sergey Bratus

https://www.nostarch.com/rootkits
THE END ! Really !?

Alexander Matrosov (@matrosov)
Eugene Rodionov (@vxradius)
Gabriel Negreira Barbosa (@gabrielnb)
Rodrigo Rubira Branco (@BSDaemon)

{alexander.matrosov || gabriel.negreira.barbosa || rodrigo.branco}
*noSPAM* intel.com
1 rodionov *noSPAM* eset.com