0X3E9 WAYS TO DIE
Who am I?

- Yaniv Balmas (@ynvb)
- Security Researcher @ Check Point Software Technologies
- Malware Research
- Vulnerability Research
- Spend most of my day staring at assembly code and binary files.
What is the problem?

- Static analysis tools contain a lot of useful data about binary files.
- Dynamic analysis tools (e.g., Debuggers) contain all execution flow related data.
- It seems trivial to bridge those two approaches.

“Well I wish you’d just tell me rather than try to engage my enthusiasm.”

– Marvin
PREVIOUS SOLUTIONS
IDA-Splode

- 2014, Zach Riggle
- Uses Intel PIN framework

- Very extensive tracing
- Branch Statistics
- Data is stored as .IDB comments
- Only works on INTEL archs and is designed mainly for Windows.
Funcap

- 2013, Andrzej Derezowski
- Uses IDA Debugging API
- Very intuitive solution
- Parses ASCII\Unicode string values
- New threads are not being followed
- Argument offsets are calculated “manually”
The extracted dynamic data is not indexed and searching through it can be a *pain*.

Entry level for adding custom functionality is relatively high.

**NO REFERENCE TO VALUE TYPES!!**
DON'T PANIC

( And Prepare to DIE... )
How to DIE?

- DIE - “Dynamic IDA Enrichment”
- Collect context from function calls & returns only.
- Parse argument values and present them in a “Human Readable” format.
- Smart interaction between static & dynamic data.
- Use as much IDA-API Magic as possible.
How to DIE?

- DIE - “Dynamic IDA Enrichment”
- Collect context from function calls & returns only.
- Parse argument values and present them in a “Human Readable” format.
- Smart interaction between static & dynamic data.
- Use as much IDA-API Magic as possible.
How can we query IDA for function argument types?
Once we have the argument values, how do we parse them? Which values should we parse?
How do we parse complex data types? (Structs, unions, pointers, etc)?
After hours of fun reading IDA-API, it turns out there are some objects we can actually use.

tinfo_t objects holds a ridiculous amount of information about data types.

Digging even deeper into tinfo_t object reveals the func_type_data_t, func_arg_t and arg_loc_t objects which store everything we need to parse function arguments.
Impossible to think of all parsing options!

Makes more sense to create an open source plugin framework.
Impossible to think of all parsing options!

Makes more sense to create an open source plugin framework.
What do we do when we encounter a complex data type?

Simple. Break it up until we reach the simple types.

**Structs / Unions**

```
udt = udt_idaapi.udt_type_data_t
```

```
type_info.get_udt_details(udt)
```

**Arrays**

```
arr = idaapi.array_type_data_t()
```

```
type_info.get_array_details(arr)
```

**References**

```
type_info.get_pointed_object()
```
VALUE PARSERS

0x486176

“Have a Nice Day!”
### Simple Value Parsers

<table>
<thead>
<tr>
<th>Parser Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>String Parser</strong></td>
<td>Uses <code>idc.GetString</code> to parse ASCII, Unicode, Pascal and other strings</td>
</tr>
<tr>
<td><strong>Bool Parser</strong></td>
<td>Returns TRUE for 0x1 and FALSE for 0x0. (Duh!)</td>
</tr>
<tr>
<td><strong>Function Parser</strong></td>
<td>Returns the referenced function name.</td>
</tr>
<tr>
<td><strong>Module Parser</strong></td>
<td>Returns the referenced module name.</td>
</tr>
</tbody>
</table>
Advanced Parser – Handles

- Works on Windows systems with local debugger (currently).
- Uses DuplicateHandle() to duplicate the handle associated with the raw value from the current running process.
- Uses NtQueryObject() on the local handle to retrieve the handles details.
- Returns the handle name and type.
Great example of an ad-hoc parser.

Check if the value pointed by offset 4 of raw address is either a string or references a string.

Also, make sure raw value is not a string.
“Demos, don't talk to me about demos…”

-Marvin
Your Orders:

Assigned By:
Agent M

Target Application:
Some Firmware Upgrade Utility

Mission:
Bypass password protection
Quickly!
Defeat C++ Code

YOUR ORDERS:

Assigned By: Agent M

Target Application: 7zip cli (32-bit version)

Mission: Get a complete code analysis Quickly!
String De-Obfuscation

YOUR ORDERS:

Assigned By: Agent M

Target Application: Explosive Trojan

Mission: Find the string de-obfuscation function

Quickly!
#TODO

- Thunk Functions
- Complex function parsers
- Better GUI
- (Much) Better DB
- Solve (very) dramatic crashes
Looks cool, Can I have it?

- Yes.
- DIE is an open source tool.
- https://github.com/ynvb/DIE
- If you like it, contribute.
SARK
IDA Python Made Easy

- Simple
- Intuitive
- Object Oriented API

Docs: sark.rtfd.org
Code: github.com/tmr232/sark
Written by Tamir Bahar @tmr232