

BYAKUGAN

眼
開
眼
白

Increase Your Sight



Byakugan

"White Eyes"

- Byah-ku-gon
 - Not Byak-ta-gon
 - Not Bye-uh-ku-gun
 - Not Four-Yaks-And-A-Dog
- A windows native debugging plugin
 - Tested on:
 - Vista
 - XPSP2

眼
開
眼
白

Functionality

- Real Time Heap Visualization
- Buffer Identification and Hunting
- Return Address Hunting
- Soon to have a lot more...
 - Metasploit / Fuzzer integration

Prerequisites

- Windows Server 2003 Driver Development Kit
- Debugging Tools For Windows (w/ SDK)
 - Install to C:\windbg\
- Detours (included pre-built)

Build Instructions

- The windbg sdk is currently busted.
- To fix it, you'll need to edit wdbgexts.h
C:\windbg\sdk\inc\wdbgexts.h

```
28: #ifndef _WDBGEXTS_  
29: #define _WDBGEXTS_  
30:  
31: #define __field_ecount_opt(x) ()  
32:  
33: #if _MSC_VER > 1000
```

Build Instructions

- Open a Windows 2003 Server Free x86 build window from the DDK
- Setup.bat
- This will build both byakugan.dll and injectsu.dll

Installation

- We just need to copy the built dlls to their needed locations:

```
copy i386\byakugan.dll C:\windbg\
```

```
copy injectsu\i386\injectsu.dll C:\windbg\
```

```
copy bin\detoured.dll C:\Windows\System32\
```

Byakugan.dll

- Interfaces with the debug API
- Defines and manages user commands
- Provides the fuzzer and process back channel listener threads
- Contains the bulk of the plugin

Injects.dll

- Handles hooking API functions in the target process
- Creates a back channel information gathering thread which connects to the debugger
- Contains all the replaced functions

Detoured.dll

- Microsoft Research hooking library
- Handles trampoline code
- Keeps track of hooked functions
- Provides auto fix-ups on function trampolines

Loading Byakugan

- Start windbg
- Attach to a process
 - Ensure the process is running at the same access level, or tenketsu will not work
- !load C:\windbg\byakugan.dll

眼
開
眼
白

Building and Loading Demo

Buffer Registration Jutsu

It means technique.

- `!jutsu identBuf MyBufName CONTENTS`
- `!jutsu identBuf msfpattern 500`
- `!jutsu listBuf`
- `!jutsu rmBuf MyBufName`

!jutsu identBuf

- Identify a buffer you've sent as input by a name and contents
- `!jutsu identBuf msfPattern 500`
Registers a metasploit pattern buffer of size 500
- `!jutsu identBuf Name Contents`
Registers the buffer of name 'Name' containing 'Contents'

!jutsu listBuf

0:000> !jutsu listBuf

[J] Currently tracked buffer patterns:

Buf: Name

Pattern: Contents

Buf: msfpattern

Pattern: Aa0Aa1Aa2Aa...

眼
開
眼
白

!jutsu rmBuf

- Remove buffers by name:

```
0:000> !jutsu rmBuf Name
```

```
[J] Removed buffer: Name
```

```
0:000> !jutsu listBuf
```

```
[J] Currently tracked buffer patterns:
```

```
Buf: msfpattern
```

```
Pattern: Aa0Aa1Aa2Aa...
```

!jutsu hunt

- The point of buffer tracking is so we can learn what buffer has gained us a crash
- We can also use buffers to find direct and indirect return addresses

0:000> !jutsu hunt

[J] Controlling eip with msfpattern at offset 332.

!jutsu findReturn

- Hunts down indirect return addresses:

```
0:000> !jutsu findReturn
```

```
[J] started return address hunt
```

```
[J] opcode test buffer starts at 0x00020000
```

```
[J] Address 0x75b28bf7 contains a call [ebp+c] instruction which will  
return to buf msfpattern at offset 0x00000148
```

```
[J] Address 0x00780853 contains a call [ebp+30] instruction which  
will return to buf msfpattern at offset 0x00000148
```

眼
開
眼
白

Buffer Jutsu Demo

!tenketsu

- Tenketsu (Ten-Ket-Sue, “Vital Points”) – Byakugan’s real time heap visualization component
- Tenketsu works by injecting a dll (injectsu) into the target process
 - Hooks heap functions in ntdll
 - Spawns a thread to read/write to a named pipe connected to the debugger

!tenketsu

- Undocumented functions are dynamically resolved from pdbs to prevent versioning issues
- Information fed back to the debugger is used to model the heap as changes occur to it

!tenketsu

- `!tenketsu`
- `!tenketsu listHeaps`
- `!tenketsu listChunks`

眼
開
眼
白



!tenketsu

- Initialize heap visualization with !tenketsu

```
0:000> !tenketsu
```

```
[Byakugan] Beginning data gathering thread... Success!
```

```
[Byakugan] Injecting Tenketsu Heap Monitoring DLL... Success!
```

```
[Byakugan] Waiting for named pipe connection...
```

```
[Byakugan] Resolving undocumented Heap functions...
```

```
[T] Resolved undocumented function
```

```
'ntdll!RtlpCoalesceFreeBlocks' @ 0x77201a25.
```

!tenketsu

- When you run the target again, the new thread will connect back, and send undocumented function information:

0:000> g

[Byakugan] Connected to back channel. :)

[T] Sending address of ntdll!RtlpCoalesceFreeBlocks

[T] Sent addresses of 1 undocumented functions.

!tenketsu listHeaps

- You can list the currently tracked heaps with `!tenketsu listHeaps`:

```
0:000> !tenketsu listHeaps
```

```
[T] Currently tracking 3 heaps:
```

```
Base: 0x003c0000    Number of Chunks: 11
```

```
Flags: 0x00000000   Reserve: 0x00000000
```

```
Lock: 0x00000000
```

```
Base: 0x00150000    Number of Chunks: 3
```

```
Flags: 0x00000000   Reserve: 0x00000000
```

```
Lock: 0x00000000
```

```
Base: 0x00360000    Number of Chunks: 13
```

```
Flags: 0x00000000   Reserve: 0x00000000
```

```
Lock: 0x00000000
```

!tenketsu listChunks

- You can list the chunks tracked per heap with
`!tenketsu listChunks heapBase`

```
0:000> !tenketsu listChunks 0x00360000
```

```
[T] Currently tracking 13 chunks for heap 0x00360000
```

```
Address: 0x00362b00      Size: 0x00000100      Flags: 0x00000000  
FREE'D
```

```
Address: 0x00362c18      Size: 0x00001000      Flags: 0x00000000  
IN USE
```

```
Address: 0x00363c30      Size: 0x00000100      Flags: 0x00000000  
IN USE
```

眼
閉
眼
白

Passive Tenketsu

- Tenketsu will point out flaws such as double frees in some cases:

[T] Possible 'double free' of chunk @
0x003643d8

眼
開
眼
白

Tenketsu Heap Visualization Demo

Tenketsu Design

Windbg

Target

Tenketsu

Injects

Heap Modeler

Named Pipe

Main Windbg Thread

- Creates the Tenketsu thread in the debugger process
- Injects the injectsu dll into the target process
- Creates an injectsu thread in the target process

Tenketsu Thread

- Creates the back channel named pipe, and waits for a connection from the target process
- Resolves undocumented functions and sends their information through the pipe
- Enters a read loop, getting information back to the target process, and sending to the heap modeler

Heap Modeler

- The heap modeler receives all information directly from the real time heap interface in ntdll
- This information is stored in a multilevel data structure as it is received
- This information can be recalled and displayed at break, or in real time by another visualization thread

Injtsu Thread

- Injtsu performs three major tasks:
 - Create a connection to the debugger through the named pipe
 - Collect information from the debugger on undocumented functions
 - Hook all Rtl functions pertaining to the heap
- These hooks call the actual rtl functionality, and pass any necessary information through the named pipe to the debugger

Further Goals

- Connect to an MSF based fuzzer
- Generate full code coverage graphs
- Generate changed memory graph
- Lots more...