BROWSER CRASH ANALYSIS

DAVID D. RUDE II

WHOAMI

WHY SHOULD YOU LISTEN TO ME?

- David D. Rude II
- Corelan Team Member
- Former iDefense Labs
- Former Metasploit Lead Exploit Developer
- Currently at FusionX LLC
- Twitter: @bannedit0
- Email: bannedit0@gmail.com

WHOAMI

ASKING QUESTIONS

- irc.freenode.org
- #g4h
- “[Q] my question is…”
- I’ll do my best to answer as many as I can at the end

INTRODUCTION

WHAT ARE WE LEARNING TODAY?

- Debugging
- Memory Management
- Use After Free
- Prototyping Exploitation

INTRODUCTION

WHY DO WE CARE ABOUT BROWSERS?

- Browsers are wide spread, everyone uses them
- Very large and complex code bases
- Handle a wide range of file formats and protocols
- New code is added regularly
- W3C compliance
- Easy to control memory
- Treasure trove for bug hunters

DEBUGGING

DEBUGGING

WINDBG BASICS

- WinDBG is a powerful kernel and userland debugger
- Memory inspection
- Hardware breakpoints
- Conditional breakpoints
- Extensible
- Python support with PYKD

DEBUGGING

MEMORY INSPECTION

- dd - display dword
- dds - display dword with symbols
- dp - display pointers
- dps - display pointers with symbols
- db - display bytes
- du - display as unicode
- da - display as ascii

DEBUGGING

HARDWARE BREAKPOINTS

- ba - break on access
- set the access mode
- ba [mode][size] address
- ba r4 address
- break on read access of 4 bytes from address
- supports R/W/X
- sizes: 1, 2, 4

DEBUGGING

CONDITIONAL BREAKPOINTS

- bp - set a breakpoint
- bp address
- bp address “j (Condition) ‘OptionalCommands’; gc”
- bp address “.if (Condition) { OptionalCommands } .else { gc }

DEBUGGING

WINDB JSCRIPT LOGGING SCRIPT

DEMO

DEBUGGING

EXTENSIONS

- WinDBG can load binary extensions
- Narly
- bNarly
- byakugan
- pykd
- Exploitable
- All the !commands

DEBUGGING

EXTENSIONS

- !heap - very useful for heap debugging
- !heap -p -a address
- Displays detailed info about a heap chunk
- !heap -v 0
- Walks the heap and validates everything
- Useful for heap corruption detection
- !heap -flt s 0x80
- Display stats about blocks which are of 0x80 in size

DEBUGGING

PYTHON SUPPORT

- Provided by pykd extension
- http://pykd.codeplex.com
- mona.py
- counterfeit.py
- example: !py mona
- write your own!

DEBUGGING

PAGE HEAP

- Heap allocation monitoring tool
- Very handy for debugging
- Can cause bugs to show up earlier rather than later
- Provides heap allocation and free stack traces +ust
- Two modes, full page heap and standard page heap

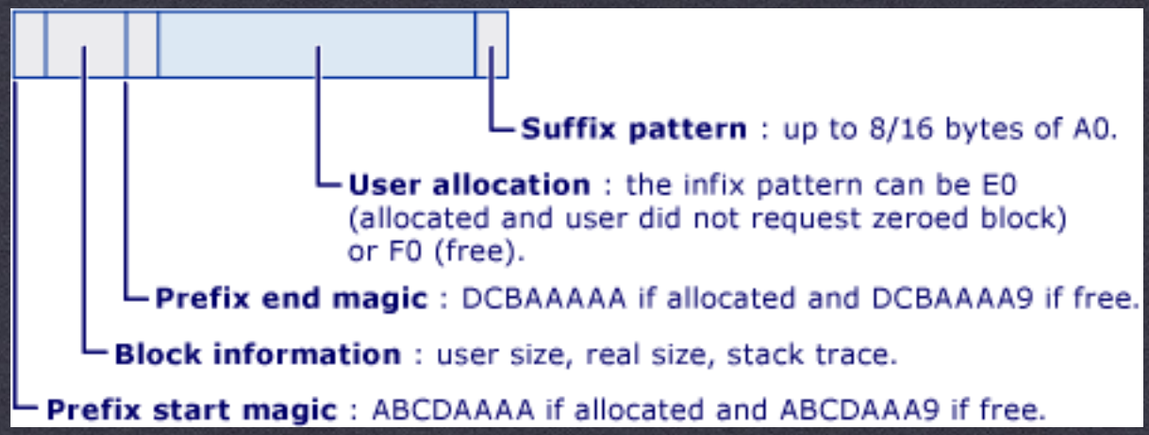
DEBUGGING

PAGE HEAP, HOW DOES IT WORK?

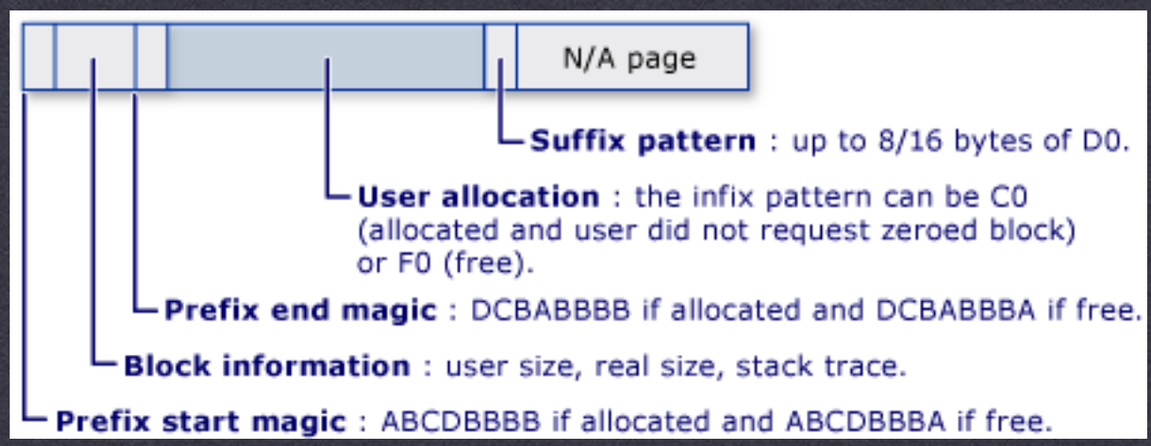
- Page heap will allocate 2 pages of memory
- The first page is given back as the user pointer
- The second page has PAGE_NOACCESS permissions
- Enables us to detect heap buffer overflows
- Ensure it is enabled inside WinDBG using !gflag

PAGE HEAP, HOW DOES IT WORK?

Standard Page Heap Header



Full Page Heap Header



DEBUGGING

PAGE HEAP, HOW DOES IT WORK?

- Page heap is not exploit friendly
- But it does allow us to track down heap bugs
- Use-After-Free
- Heap Buffer Overflow (even off-by-ones)

DEBUGGING

PAGE HEAP, HOW DOES IT WORK?

- gflags.exe /i iexplore.exe +hpa +ust
- enable pageheap on iexplore.exe
- !gflag
- ensure pageheap is enabled in WinDBG

DEBUGGING

WINDB REFERENCES

MSDN:

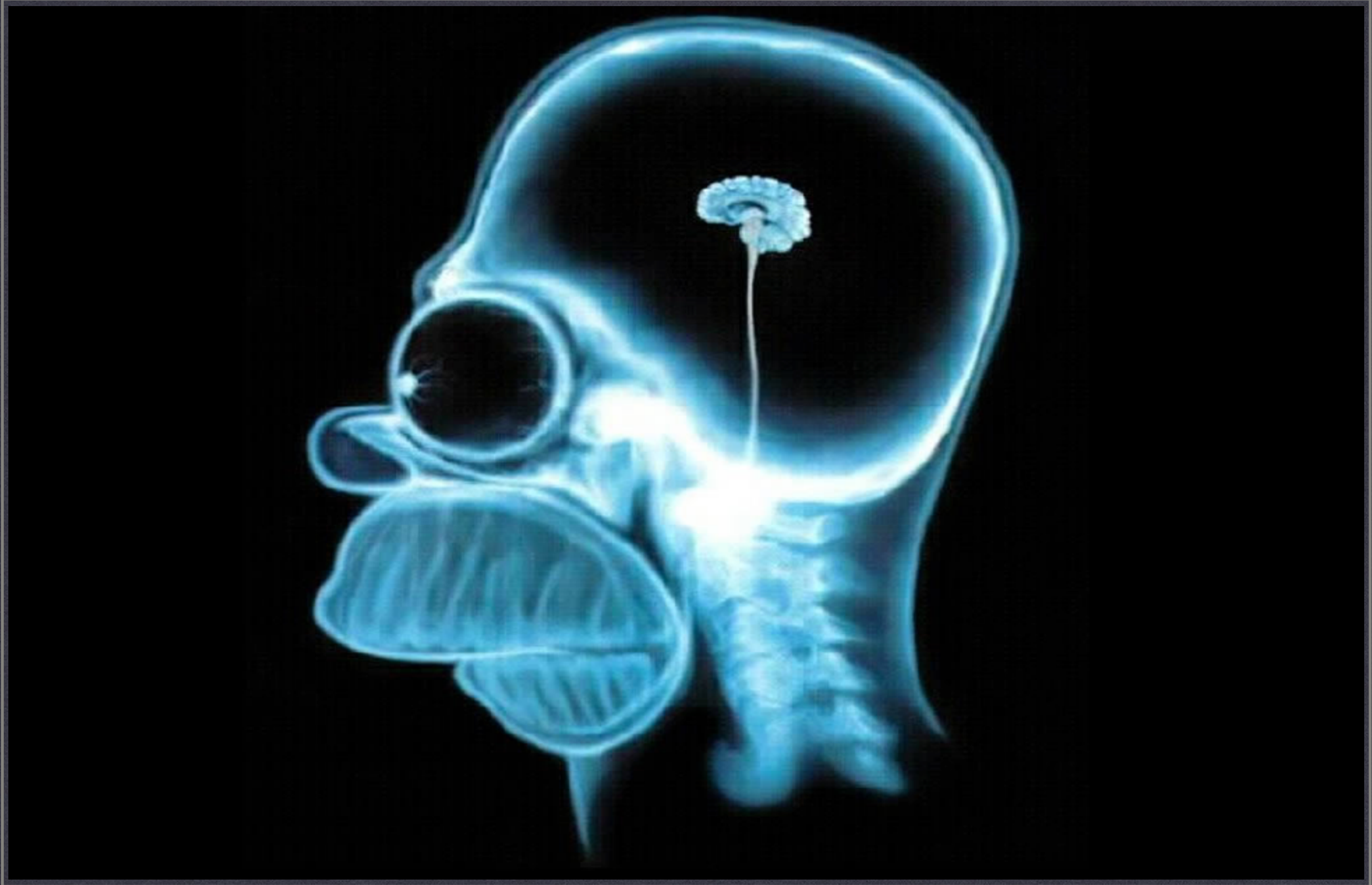
http://msdn.microsoft.com/en-us/library/windows/hardware/ff539170(v=vs.85).aspx

Cheat Sheet:

http://windbg.info/doc/1-common-cmds.html

General:

http://www.windbg.org/



MEMORY MANAGEMENT

MEMORY MANAGEMENT

IT REALLY IS IMPORTANT TO UNDERSTAND

- Exploitation requires an understanding of MM
- Many different heap allocators for various browsers
- IE has a fairly well explored behavior
- Heap Feng Shui

MEMORY MANAGEMENT

ALLOCATIONS

- Allocations can be handled in two ways
- Handled by either the Front End or Back End allocator
- Front End is used for requests under 16kb
- Front End is not active initially
- The Front End is called Low Fragmentation Heap (LFH)
- Front End is enabled for size bins after several requests
- LFH enabled for a size after 18 requests for that size
- If LFH is not enabled for a size the Back End is used

MEMORY MANAGEMENT

FREE

- For exploitation we primarily care about LFH behavior
- Freeing memory with LFH enabled is deterministic
- LIFO
- Meaning, we can allocate memory, free it, and allocate it again
- Using this we can craft the heap into ideal layouts for exploits

MEMORY MANAGEMENT

LFH IN ACTION

LFH Bin 512

Initial state of the heap

MEMORY MANAGEMENT

LFH IN ACTION

LFH Bin 512

Allocate memory →

mem = HeapAlloc(512)

MEMORY MANAGEMENT

LFH IN ACTION

Allocate memory →

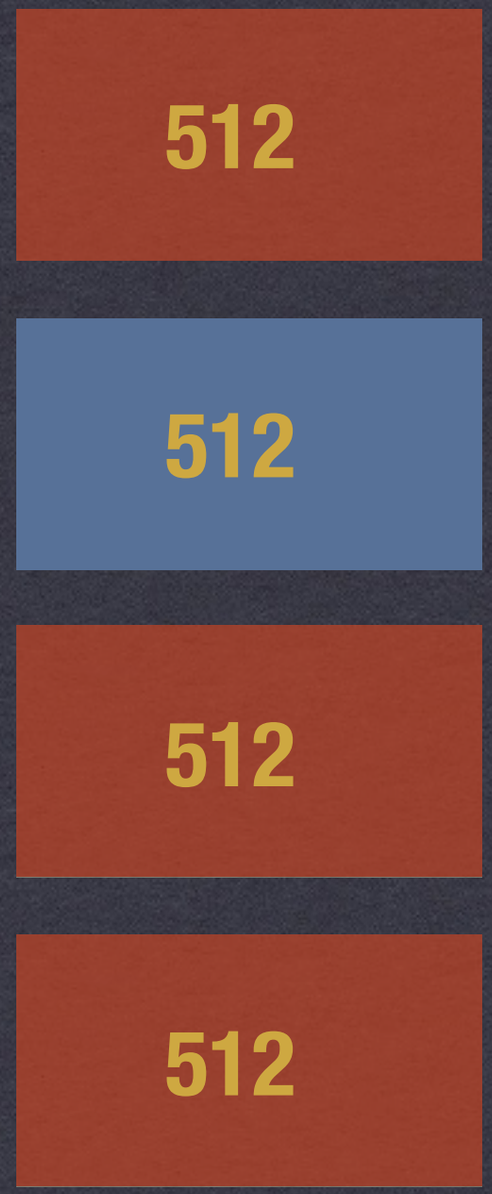
mem2 = HeapAlloc(512)

MEMORY MANAGEMENT

LFH IN ACTION

Free memory →

HeapFree(mem)



MEMORY MANAGEMENT

LFH IN ACTION

Allocate memory →

mem3 = HeapAlloc(512)

mem3 just filled the hole left by freeing mem

MEMORY MANAGEMENT

LFH IN ACTION

- Using LFH we can free blocks of a specific size
- Cause objects of the same size to take up the holes
- With this we can craft the heap as we please
- This is useful for memory disclosures ;)
- Harder on Windows 8 ;(

MEMORY MANAGEMENT

ENABLING LFH

DEMO

MEMORY MANAGEMENT

MM REFERENCES

http://media.blackhat.com/bh-us-12/Briefings/Valasek/BH_US_12_Valasek_Windows_8_Heap_Internals_Slides.pdf

http://illmatics.com/Understanding_the_LFH_Slides.pdf

https://www.youtube.com/watch?v=s7sF5arWWcU

https://www.lateralsecurity.com/downloads/hawkes_ruxcon-nov-2008.pdf

https://www.blackhat.com/presentations/bh-europe-07/Sotirov/Presentation/bh-eu-07-sotirov-apr19.pdf

USE AFTER FREE

USE AFTER FREE

UAF INTRODUCTION

- Caused by freeing memory and then referencing later
- Can be hard to track down
- PageHeap helps a lot
- PageHeap lets us see the stacktrace of the free call
- Typically seen in complex C++ Object Oriented code
- Can also occur in C malloc/free

USE AFTER FREE

UAF PROTECTIONS?

- Microsoft has added VTGuard to IE 10 and 11
- Checks to see if a pointer is intact before calling virtual functions
- Not all classes are protected currently
- Can be bypassed with a memory disclosure of MSHTML (calculate the vtguard pointer)
- Can also be avoided in some cases

USE AFTER FREE

UAF EXPLOITATION

- Only possible if we can control the object that is freed
- Using LFH we reallocate the same memory location
- This means we need to make an allocation of the same size as the object

USE AFTER FREE

CVE-2014-0322

DEMO

PROTOTYPING EXPLOITATION

PROTOTYPING EXPLOITATION

WHAT CAN WE ALREADY DO?

- We know we can do certain things from a browser
- We can cause allocations
- We can even cause frees
- That is all we need for prototyping exploitation

PROTOTYPING EXPLOITATION

CAVEATS

- Can we control the object from the browser?
- Not all code paths will lead to a write
- A write might never occur
- No write, means its likely not exploitable
- Test all the code paths, we control conditionals

PROTOTYPING EXPLOITATION

HOW DO WE PROTOTYPE?

- We can allocate memory in the target process
- Write to that memory
- This allows us to create fake objects in memory
- Trigger a Use-After-Free
- Set the object reference to our fake object
- Step forward through the code and see what happens

PROTOTYPING EXPLOITATION

INTRODUCING COUNTERFEIT

- Counterfeit is a pykd script
- Allocate memory in the target process
- Fine graned page permission control
- Allocate R, RW, RWX, NOACCESS
- Also, allows for page fault breakpoints

PROTOTYPING EXPLOITATION

INTRODUCING COUNTERFEIT

DEMO

THE END

Questions?