EXPLOITING
NATIVE
CLIENT

- BEN HAWKES
THE INTRODUCTION

• ben
THE INTRODUCTION

• ben
• + mark
Announcing the Native Client Security Contest

By Henry Bridge, Native Client Team

Exploits, bugs, vulnerabilities, security holes -- for most programmers these terms are synonymous with fire drills and coding all-nighters. However, for the next 10 weeks, the Native Client team is inviting you to bring them on! We're challenging you to find security exploits in Native Client. Sign up today for the Native Client Security Contest, you could win up to $213, as well as recognition from renowned security researchers.

Before getting started, you must complete the registration process for yourself or your team. Then, you can grab the latest build of Native Client, attack it to find security holes, and submit the ones you discover. You get credit for bugs that your team reports first. If another contestant submits a vulnerability before you, or we publish a fix before you report it, well then... you'll have to keep looking!

At the end of the contest, all entries will be reviewed by a panel of academic experts, chaired by Edward Felten of Princeton University. They will select the five eligible entries with the most high-impact bugs, and these winners will receive cash prizes, as well as anam bragging rights. For more details, please review the contest's terms and conditions.

Registration is now open and the contest will run until May 5th. Sign up today to start reporting exploits as soon as possible.

Happy bug hunting!

Labels: native client
THE INTRODUCTION

• ben
• + mark
• = beached as
THE INTRODUCTION

“Native Client is an open-source research technology for running x86 native code in web applications, with the goal of maintaining the browser neutrality, OS portability, and safety that people expect from web apps.”
THE INTRODUCTION

“Native Client is an open-source research technology for running x86 native code in web applications, with the goal of maintaining the browser neutrality, OS portability, and safety that people expect from web apps.”

• x86 code delivered to client browser from remote server (web app)
  • this code must work on any browser on any OS
    • and be run in such a way that is “secure”

Definitions of insanity on the Web:

• relatively permanent disorder of the mind
  wordnetweb.princeton.edu/perl/webwn

• Traditionally, insanity, craziness or madness is the behavior whereby a person flouts societal norms and may become a danger to themselves and …
  en.wikipedia.org/wiki/Insanity
THE INTRODUCTION

Schedule:
• technical kung-fu

• some speculative corporate analysis

• parting remarks + questions/discussion
TECH
THE GOAL

Motivation: break the native client security model
THE GOAL

Motivation:
break the native client security model

but what is the security model?
THE METHOD

The Common Sense Methodology:
- understand the design
- understand the code
- audit
- test
- audit
- test
- ....
NATIVE CLIENT TECHNOLOGY

WEB SERVER

BROWSER

NACL BROWSER PLUGIN

SERVICE RUNTIME

NEXE
NATIVE CLIENT TECHNOLOGY

SERVICE RUNTIME PROCESS

untrusted i386 cs/ds/ss segment

TRAMPOLINE
ELF
NEXE

loader
validator
imc
syscalls
etc...
// TextLimit = the upper text address limit
// Block(IP) = 32-byte block containing IP
// StartAddr = list of inst start addresses
// JumpTargets = set of valid jump targets

// Part 1: Build StartAddr and JumpTargets
IP = 0; icount = 0; JumpTargets = { }
while IP <= TextLimit:
    if inst_is_disallowed(IP):
        error "Disallowed instruction seen"
    StartAddr[icount++] = IP
    if inst_overlaps_block_size(IP):
        error "Block alignment failure"
    if inst_is_indirect_jump_or_call(IP):
        if !is_2_inst_nacl_jmp_idiom(IP) or
           icount < 2 or
           Block(StartAddr[icount-2]) != Block(IP):
            error "Bad indirect control transfer"
    else
        // Note that indirect jmps are inside
        // a pseudo-inst and bad jump targets
        JumpTargets = JumpTargets + { IP }
        // Proceed to the fall-through address
        IP += InstLength(IP)

// Part 2: Detect invalid direct transfers
for I = 0 to length(StartAddr)-1:
    IP = StartAddr[I]
    if inst_is_direct_jump_or_call(IP):
        T = direct_jump_target(IP)
        if not(T in [0:TextLimit])
           or not(T in JumpTargets):
            error "call/jmp to invalid address"

Figure 3: Pseudo-code for the NaCl validator.
1. Disassemble binary, invalidate (exit!) on “dangerous” instructions
2. Invalidate on instructions straddling blocks (i.e. block unaligned)
3. For indirect branches, ensure block alignment primitive used on target
4. Record list of properly aligned “valid” branch targets
5. Restart disassembly from start to check all branches hit valid targets
The validator comes down to this:

- if your instructions are good

- and you branch to instructions

then its all good mate
INITIAL ATTACKS

An initial attack surface:

- browser plugin
- binary loader
- nexe validator
- runtime services
Native client is C/C++

this is essentially required

“its like 1999”
Native client is C/C++

this is essentially required

“its like 1999”

DEMONSTRATION!
Beached As founds bugs in:

- validator
- syscall
- imc
- browser plugin
SRPC Shared Memory Infoleak / Memory Corruption

browser plugin integer overflow

visit a website ------>

arbitrary code execution in your browser
bool SharedMemory::Invoke(...) {

    uint32_t offset;
    uint32_t len;

    offset = NPVARIANT_TO_INT32(args[0]);
    len = NPVARIANT_TO_INT32(args[1]);

    if (offset + len > shared_memory->size_) {
        return false;
    } else {
        char* ret_string = NPN_MemAlloc(2 * len);
        unsigned char* shm_addr = (shared_memory->map_addr_) + offset;
        for (unsigned int i = 0; i < len; ++i) {
            unsigned char c = *shm_addr;
            *out = c;
            ++out;
            ++shm_addr;
        }

        STRINGN_TO_NPVARIANT(ret_string, ..., *result);
        return true;
    }
}
SRPC Type Confusion Memory Corruption Attack

plugin compromise

classic dowd
please give me a string for the src parameter
please give me a string for the src parameter

ok here you are

`nacl.src = "have a nice day"`
please give me a string for the src parameter
please give me a string for the src parameter

ok wrrd..
nacl.src = 0x12345678
RIP 2008-2009

native client browser plugin

access violation when writing to memory:
0x12345678
2-byte Jump Operand Prefix Vulnerability

validator disassembler logic flaw

i386 instruction prefixes
“modify” instruction that follows
Nacl validator checked prefix for 1-byte branches
NaCl validator checked prefix for 1-byte branches

... but there exist 2-byte branches
Nacl validator checked prefix for 1-byte branches

… but there exist 2-byte branches

“conditional jumps”

modify code segment of a jCC
= jump anywhere into service runtime!
Direction Flag Sandbox Bypass

validator logic flaw …

leads to mem corruption in service runtime code exec in runtime process!
EFLAGS register = flags (mostly status)

- can set from inside inner sandbox

- but is NOT cleared when exe trampolines to service runtime...
Welcome to the Bizarro World

That memcpy you thought was going forwards?

Not so much.
Welcome to the Bizarro World

That memcpy you thought was going forwards?

Not so much.

“setting the DF flag causes string instructions to auto-decrement”
Native Client Memory Unmapping Vulnerability

runtime services fail

syscalls
- munmap
- mmap
Native Client Memory Unmapping Vulnerability

runtime services fail

- munmap
- mmap

need i say more?
• ELF is hard; loader bugs
• Side channels.. I guess
• CPU erratta
  Remote hardware exploits
• Inter-module exploitation
questions?

Q?

Q? Q?
THE HARD STUFF ($)
I have a question.

Can native client win?
I have a question.

Can native client win?

Technically, commercially
TARGET

Confused target audience?

Not with Chrome OS

Chrome OS = context for everything
Native Client Security Contest: The results are in!
Tuesday, July 07, 2009

A few months ago, we challenged you to discover exploits in the Native Client system and more than 600 of you decided to take us up on our invitation. We're very pleased with the results: participants found bugs that enabled some really clever exploits, but nothing that pointed to a fundamental flaw in the design of Native Client. Our judges reviewed all entries very carefully and have selected five teams as the winners of the Native Client Security Contest.

Introducing the Google Chrome OS
7/07/2009 09:37:00 PM
It's been an exciting nine months since we launched the Google Chrome browser. Already, over 30 million people use it regularly. We designed Google Chrome for people who live on the web — searching for information, checking email, catching up on the news, shopping or just staying in touch with friends. However, the operating systems that browsers run on were designed in an era where there was no web. So today, we're announcing a new project that's a natural extension of Google Chrome — the Google Chrome Operating System. It's our attempt to re-think what operating systems should be.
Native Client Security Contest: The results are in!
Tuesday, July 07, 2009

A few months ago, we challenged you to discover exploits in the Native Client system and more than 600 of you decided to take us up on our invitation. We're very pleased with the results: participants found bugs that enabled some really clever exploits, but nothing that pointed to a fundamental flaw in the design of Native Client. Our judges reviewed all entries very carefully and have selected five teams as the winners of the Native Client Security Contest.

Introducing the Google Chrome OS
7/07/2009 09:37:00 PM

It's been an exciting nine months since we launched the Google Chrome browser. Already, over 30 million people use it regularly. We designed Google Chrome for people who live on the web — searching for information, checking email, catching up on the news, shopping or just staying in touch with friends. However, the operating systems that browsers run on were designed in an era where there was no web. So today, we're announcing a new project that's a natural extension of Google Chrome — the Google Chrome Operating System. It's our attempt to re-think what operating systems should be.
Microsoft’s Steve Ballmer on Chrome OS:

"The last time I checked you don't need two client operating systems."

“There’s good data that actually says about 50% of the time someone is on their PC they’re not doing something in the web browser”
THE COMPETITION

CONCLUSION:

google should be very worried about amazon
TECH = $

Technical limitations:

no 64-bit (do you care?)

slightly decreased performance

* we will find more bugs *
API/syscall “outer sandbox” limitations

What is an NEXE allowed to do?

Not much? No killer apps.
Too much? No security.
“The inability to deliver a secure implementation is an architectural flaw.”

- Dave Aitel, Immunity kingpin

Everyone welcome Native Client to the “Advisory Treadmill”.

THE TARGET

Beware of alienating target audience with security considerations

Google Omaha ++

Defense in depth is REQUIRED
THE POINT

Everyone has the “implementation problem”

The inner sandbox is not yet broken

Native Client + Chrome OS “makes sense”
sshhh.. someone might hear

ok, this is my tentative endorsement that, yes, native client could actually win ***

*** but only if they lock tavis ormandy in a room for a year or two

... and im worried about that outer sandbox, so er, you should be too
THE END

thanks

twitter.com/benhawkes