Put in one bug and pop out more: An effective way of bug hunting in Chrome

Rong Jian, Leecraso, Guang Gong
Alpha Lab, 360 Internet Security Center

About 360 Alpha Lab

360 Alpha Lab

- More than 400 vulnerabilities acknowledged by top vendors
- Won the highest reward
  - in the history of the ASR program in 2017
  - in the history of Google VRP in 2019
- Successful pwner of several Pwn2Own and Tianfu Cup events

Introduction

Variant Analysis

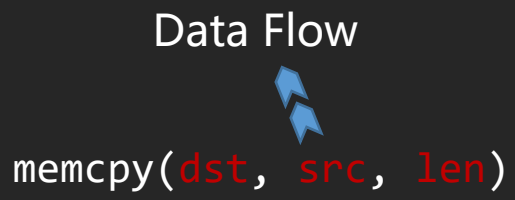
Find similar vulnerabilities based on a known one

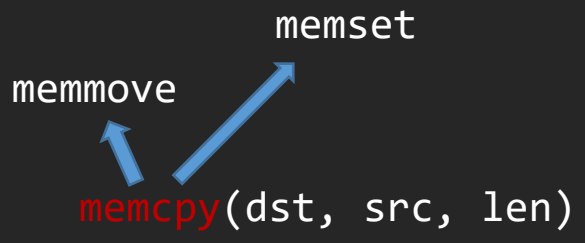
- Manual code audit
- Static analysis tools
- Fuzzing as a "seed"

Variant Analysis

Find similar vulnerabilities based on a known one

- Manual code audit
- Static analysis tools
- Fuzzing as a "seed"





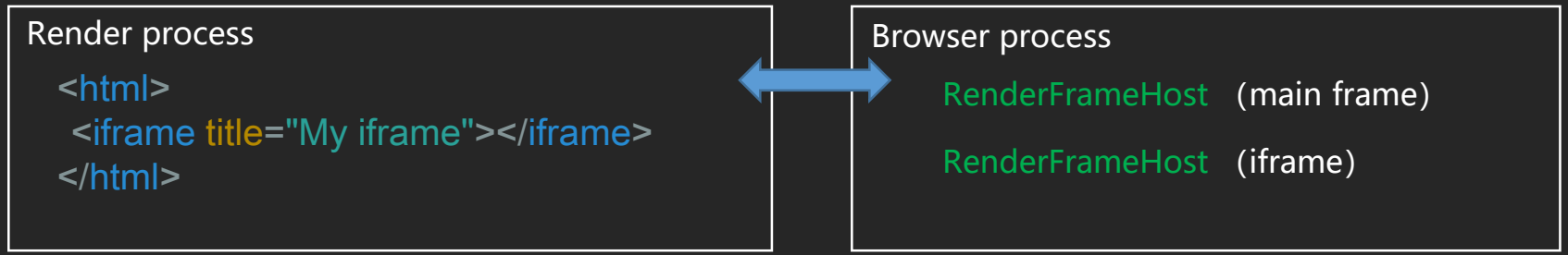
The Target : Chrome

- Chrome has a multi-process architecture
  - Focusing on the code runs in Browser process
  - Not sandboxed
- CodeQL
  - A great analysis tool
  - Compiles code to a snapshot database and
  - Can run queries against it for program analysis

RenderFrameHost Issues

RenderFrameHost (RFH)

- Lives in the browser process
- Provides a communication conduit with a frame in the render process
- Destroyed when the frame is closed



How to Access a RFH ?

- 🙂 Store a *GlobalFrameRoutingId* and using RenderFrameHost::FromID() to retrieve it back
- 😐 Hold a raw pointer to RFH

An Example

Chrome Issue 1062091

- InstalledAppProviderImpl provides installed app information related to the origin of the requesting page

```
1  class InstalledAppProviderImpl :
2                      public blink::mojom::InstalledAppProvider
3  {
4   // ...
5   private:
6     RenderFrameHost* render_frame_host_;
7  };
```

An Example

Chrome Issue 1062091

- InstalledAppProviderImpl outlives RFH
- UAF occurs after RFH deconstruction

```
1  void InstalledAppProviderImpl::Create(
2      RenderFrameHost* host,
3      mojo::PendingReceiver<blink::mojom::InstalledAppProvider>
receiver) {
4      mojo::MakeSelfOwnedReceiver(
5          std::make_unique<InstalledAppProviderImpl>(host),
6          std::move(receiver));
7  }
```

Finding Bug Variants

Candidates

- Strore RFH as raw pointer in a member variable

Finding Bug Variants

Reduce false positives

WebContentsObserver

- Class can get notified of page events by inheriting it
- Give a chance to clean up when RFH is going away

FrameServiceBase

- Wrapper class of WebContentsObserver
- Work the same way

Finding Bug Variants

Candidates

- Strore RFH as raw pointer in a member variable
- Not a subclass of FrameServiceBase
- Not a subclass of WebContentsObserver or the RenderFrameDeleted method is not implemented

Finding Bug Variants

CodeQL query

```
class ClassContainsRFHPtr extends Class {
  ClassContainsRFHPtr() {
    exists(Field field | this = field.getDeclaringType()
      and (
        field.getType().getName().matches("%RenderFrameHost%*%")
      )
    )
    and not
    this.getABaseClass().getName().matches("FrameServiceBase")
    and not
    this.getAMemberFunction().getName().matches("RenderFrameDeleted")
  }
 }
```

Finding Bug Variants

23	D:/chromium/src/components/content_capture/browser/content_capture_receiver.h	ContentCaptureReceiver
24	D:/chromium/src/components/printing/browser/print_manager.cc	FrameDispatchHelper
25	D:/chromium/src/content/browser/frame_host/render_frame_host_impl.h	RenderFrameHostImpl
26	D:/chromium/src/content/browser/frame_host/frame_tree_node.h	FrameTreeNode
27	D:/chromium/src/content/browser/frame_host/frame_tree.h	FrameTree
28	D:/chromium/src/content/browser/frame_host/navigation_request.h	NavigationRequest
29	D:/chromium/src/content/browser/frame_host/raw_clipboard_host_impl.h	RawClipboardHostImpl

RawClipboardHostImpl

Reported as issue 1117348

ERROR RETURN ISSUES

RenderFrameHost lifetime issue is a too common.

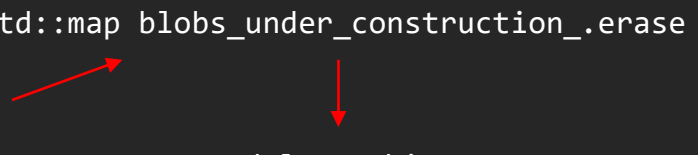
The way how to mutate the pattern is important.

ERROR RETURN ISSUES

=> found 14 vulnerabilities and got 5 CVEs

Example - CVE-2020-6461

```
1  void BlobRegistryImpl::BlobUnderConstruction::TransportComplete(
2   …
3   if (context()->registry().HasEntry(uuid())) {
4    if (result == BlobStatus::DONE)
5      context()->NotifyTransportComplete(uuid());
6    else
7      context()->CancelBuildingBlob(uuid(), result);
8   }
9   if (BlobStatusIsBadIPC(result)) {
10    std::move(bad_message_callback_)
11        .Run("Received invalid data while transporting blob");
12  }
13  …
```



Example - crbug/1065704

```
1  void WebSocket::ReadAndSendFromDataPipe() {
2   …
3    const size_t size_to_send =
4        std::min(static_cast<uint64_t>(readable_size), data_frame.data_length);
5    auto data_to_pass = base::MakeRefCounted<net::IOBuffer>(size_to_send);
6    const bool is_final = (size_to_send == data_frame.data_length);
7    memcpy(data_to_pass->data(), buffer, size_to_send);
8    channel_->SendFrame(is_final, MessageTypeToOpCode(data_frame.type),
9                        std::move(data_to_pass), size_to_send);
10
11   const MojoResult end_result = readable_->EndReadData(size_to_send);
12   DCHECK_EQ(end_result, MOJO_RESULT_OK);
13  …
14 }
```

Root cause :

During the code execution of a class instance, calling other function which could cause the destruction of this instance.

The UAF will occur if any member variable or member function is accessed after that.

```
class A {
 void A::Func(){
  …
  Foo();
  …
  mem_var_->DoSth();
 }
}
```

map.erase

```
1   ...
2   key_map.getType().stripType() instanceof ManagedMapType and
3   key_map.getType().stripType().(ManagedMapType).getManagedType() =
map_field.getManagedType() and
4   reset_func = map_field.getAManagedReset().getEnclosingFunction() and
5   reach*(ext_func, reset_func) and
6   member_f1.getDeclaringType() = map_field.getManagedType() and
7   fc.getTarget() = ext_func and
8   fc.getEnclosingFunction() = member_f1 and
9
10  (member_V.getAnAccess() = ex and
11     ex.getEnclosingFunction() = member_f1)
12  or
13  (member_fc.getTarget() = member_f2 and
14     member_fc.getEnclosingFunction() = member_f1) and
15  ...
```

CVE-2021-21115

```
1  void PasswordProtectionRequest::OnWhitelistCheckDone(bool match_whitelist) {
2  DCHECK(CurrentlyOnThread(ThreadID::UI));
3  if (match_whitelist) {
4    if (password_protection_service_->CanSendSamplePing()) {
5      FillRequestProto(/*is_sampled_ping=*/true);
6    }
7    Finish(RequestOutcome::MATCHED_WHITELIST, nullptr);
8  } else {
9   StartTimeout();
10   CheckCachedVerdicts();
11  }
12}
```

```
1  void PasswordProtectionRequest::SendRequest() {
2  DCHECK(CurrentlyOnThread(ThreadID::UI));
3
4   web_ui_token_ =
5      WebUIInfoSingleton::GetInstance()->AddToPGPings(*request_proto_);
6
7   std::string serialized_request;
8   if (!request_proto_->SerializeToString(&serialized_request)) {
9     Finish(RequestOutcome::REQUEST_MALFORMED, nullptr);
10    return;
11  }
12 …
13}
```

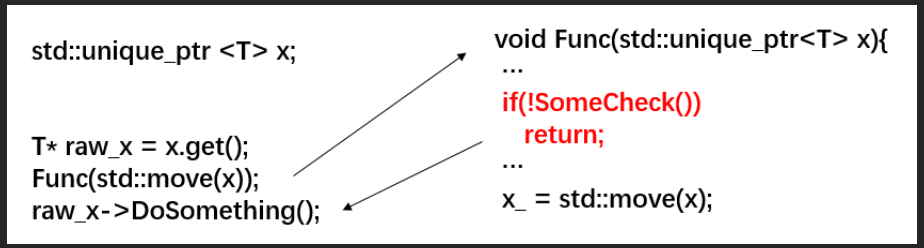
delete the request instance

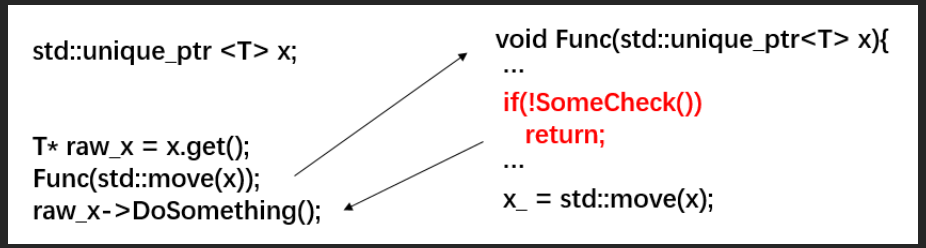
std::set pending_requests_.erase

how to "mutate"

Releasing objects in some unexpected conditional branches is prone to be vulnerable.

Focus on lifetime management of the object referenced by a smart pointer.






```
1  ...
2  and fc.getTarget() = vuln_func
3  and fc.getAnArgument() = pass
4  and vuln_func = para.getFunction()
5  and move.getEnclosingFunction() = vuln_func
6  and move.getTarget() instanceof StdMove
7  and move.getAnArgument() = para.getAnAccess()
8  and ret.getEnclosingFunction() = vuln_func
9  and not dominates(move,ret)
10 ...
```

crbug.com/1150328

```
1  void DistillCurrentPageAndView(content::WebContents* old_web_contents) {
2   …
3   std::unique_ptr<content::WebContents> old_web_contents_owned =
4      CoreTabHelper::FromWebContents(old_web_contents)
5          ->SwapWebContents(std::move(new_web_contents), false, false);
6
7   std::unique_ptr<SourcePageHandleWebContents> source_page_handle(
8      new SourcePageHandleWebContents(old_web_contents_owned.release(), true));
9
10  MaybeStartDistillation(std::move(source_page_handle));
11
12 #if !defined(OS_ANDROID)
13  dom_distiller::UMAHelper::LogTimeOnDistillablePage(old_web_contents);
14 #endif
15 }
```

```
1  void MaybeStartDistillation(
2      std::unique_ptr<SourcePageHandleWebContents> source_page_handle) {
3  const GURL& last_committed_url =
4      source_page_handle->web_contents()->GetLastCommittedURL();
5  if (!dom_distiller::url_utils::IsUrlDistillable(last_committed_url))
6      return;
7  …
8  std::unique_ptr<DistillerPage> distiller_page =
9     dom_distiller_service->CreateDefaultDistillerPageWithHandle(
10         std::move(source_page_handle));
11 …
12 }
```

The return conditions of most results is hard to be met

How to further mutate the pattern?

Found a special case

```
1  void TabStrip::TabDragContextImpl::ContinueDrag(views::View* view, const
ui::LocatedEvent& event) {
2    if (drag_controller_.get() &&
3        drag_controller_->event_source() == EventSourceFromEvent(event)) {
4      gfx::Point screen_location(event.location());
5      views::View::ConvertPointToScreen(view, &screen_location);
6      drag_controller_->Drag(screen_location);
7    }
8
9    // Note: |drag_controller| can be set to null during the drag above.
10    if (drag_controller_ && drag_controller_->group())
11      tab_strip_->UpdateTabGroupVisuals(*drag_controller_->group());
12  }
```

In Linux, Drag will eventually call X11WholeScreenMoveLoop::RunMoveLoop, which will run a nested message loop

```
1  bool X11WholeScreenMoveLoop::RunMoveLoop(
2   …
3   in_move_loop_ = true;
4   canceled_ = false;
5   base::RunLoop run_loop(base::RunLoop::Type::kNestableTasksAllowed);
6   quit_closure_ = run_loop.QuitClosure();
7   run_loop.Run();
11  …
12 }
```

Nested message loop

```
1  bool X11WholeScreenMoveLoop::RunMoveLoop(
2   …
3   in_move_loop_ = true;
4   canceled_ = false;
5   base::RunLoop run_loop(base::RunLoop::Type::kNestableTasksAllowed);
6   quit_closure_ = run_loop.QuitClosure();
7   run_loop.Run();
11  …
12 }
```

Create a new message loop

Destroy instances in new loop

Exit the new loop

Back to the blocking context and continue code execution

Save the context and create a new message loop in the current thread

Back to the special case

```
1  void TabStrip::TabDragContextImpl::ContinueDrag(views::View* view, const
ui::LocatedEvent& event) {
2    if (drag_controller_.get() &&
3        drag_controller_->event_source() == EventSourceFromEvent(event)) {
4      gfx::Point screen_location(event.location());
5      views::View::ConvertPointToScreen(view, &screen_location);
6      drag_controller_->Drag(screen_location);
7    }
8
9    // Note: |drag_controller| can be set to null during the drag above.
10    if (drag_controller_ && drag_controller_->group())
11      tab_strip_->UpdateTabGroupVisuals(*drag_controller_->group());
12  }
```

back to the context

destroy TabStrip::TabDragContextImpl in new message loop

```
1  bool X11WholeScreenMoveLoop::RunMoveLoop(
2   …
3   in_move_loop_ = true;
4   canceled_ = false;
5   base::RunLoop run_loop(base::RunLoop::Type::kNestableTasksAllowed);
6   quit_closure_ = run_loop.QuitClosure();
7   run_loop.Run();
11  …
12 }
```

CVE-2020-16004

Nested message loop Results

8 UAF bugs about the Linux X11 clipboard

2 UAF issues related to the messageBox

a series of issues in the ozone clipboard

crbug.com/1161147

```
1  void ChromePasswordManagerClient::OnPaste() {
2   …
3   if (!used_crosapi_workaround) {
4     ui::Clipboard* clipboard = ui::Clipboard::GetForCurrentThread();
5     ui::DataTransferEndpoint data_dst = ui::DataTransferEndpoint(
6         ui::EndpointType::kDefault, /*notify_if_restricted=*/false);
7     clipboard->ReadText(ui::ClipboardBuffer::kCopyPaste, &data_dst, &text);
8   }
9   was_on_paste_called_ = true;
10  password_reuse_detection_manager_.OnPaste(std::move(text));
11 }
```

back to the context

destroy ChromePasswordManagerClient in new message loop

```
1  void SelectionRequestor::BlockTillSelectionNotifyForRequest(Request* request) {
2   …
3   base::RunLoop run_loop(base::RunLoop::Type::kNestableTasksAllowed);
4   request->quit_closure = run_loop.QuitClosure();
5   run_loop.Run()
6   …
7  }
```

WeakPtr Optimization

WeakPtr

- Chromium implements its own version of weak_ptr
- Widely used in the codebase
- Null-test before accessing the underlying object

```
1 class Foo { ... };
2 WeakPtr<Foo> foo;
3 if (foo)              // null-test for validation
4   foo->method();
```

What if there is no null-test ?

Null pointer dereference ?

```
1  template <typename T>
2  class WeakPtr : public internal::WeakPtrBase {
3  // ...
4   T* operator->() const {
5    DCHECK(ref_.IsValid());
6    return get();
7   }
8
9   T* get() const {
10    return ref_.IsValid() ? reinterpret_cast<T*>(ptr_) : nullptr;
11  }
12
13 };
```

```
class Bar {
public:
  virtual void increase() { count_++; }
private:
  int count_ = 0;
};

class Foo {
public:
  Foo() {
    inner_ = new Bar();
    is_valid_ = true;
  }
  Bar* get() { return is_valid_ ? inner_ : nullptr; }
  void invalidate() { is_valid_ = false; }
private:
  Bar* inner_;
  bool is_valid_;
};

int main(){
  Foo* foo = new Foo();
  foo->invalidate();
  for(int i = 0; i < 2; i++)
    foo->get()->increase();
  cout << "Not crash" << endl;
  return 0;
}
```

Just like what WeakPtr does

```
→  test clang++ null_dref.cc -o bin
→  test ./bin
[1]    1877653 segmentation fault (core dumped)  ./bin
→  test clang++ null_dref.cc -O3 -o bin
→  test ./bin
Not crash
```

- get( ) should return null after invalidation
- Virtual function call on nullptr should crash the process

WeakPtr Optimization

~~Null pointer dereference~~

- Compiler chooses to fold the branch
- Convert a null pointer dereference crash to an exploitable UAF bug

Finding Bug Variants

Candidates

- WeakPtr as class member variable
- Being accessed without any null-test

Finding Bug Variants

Step 1 : Find function calls like *foo->method()*

CodeQL

```
1 weak_ptr.getType().getName()
2   .matches("%WeakPtr<%>%")
2 and fc.getQualifier()
5   = weak_ptr.getAnAccess().(Expr)
3 and fc.getTarget().getName()
4   .matches("%operator->%")
```

C++ Code

```
foo->method();
```

Finding Bug Variants

Step 2 : There is no null-test before accessing WeakPtr

No related to the WeakPtr

CodeQL

```
1 weak_ptr.getType().getName()
2   .matches("%WeakPtr<%>%")
2 and fc.getQualifier()
5   = weak_ptr.getAnAccess().(Expr)
3 and fc.getTarget().getName()
4  .matches("%operator->%")
5 not exists(IfStmt if_stmt |
6  isWeakPtrCheck(if_stmt, weak_ptr)
7  and dominates(if_stmt.getCondition(), fc)
8 )
```

C++ Code

```
if(condition) {
  //...
  return;
}
foo->method();
```

Results

About 363 results， analysis part of them...

- CVE-2020-15996 , CVE-2020-16014 , CVE-2020-16016
- Fixed in CL https://crrev.com/816701
- We exploited one of them to escape Chrome sandbox in TianFu Cup 2020

THE EOP VULNERABILITY

Prior Knowledge

- Mojo IPC
- Legacy IPC
  - control message
  - route message

Renderer-side send PpapiHostMsg_FileIO_Open

```
Call<PpapiPluginMsg_FileIO_OpenReply>(BROWSER,
    PpapiHostMsg_FileIO_Open(
        file_ref,
        open_flags),
    base::Bind(&FileIOResource::OnPluginMsgOpenFileComplete, this,
               callback));
```

Browser-side implement Open listener

```
int32_t PepperFileIOHost::OnHostMsgOpen(
    ppapi::host::HostMessageContext* context,
    PP_Resource file_ref_resource,
    int32_t open_flags) {
  int32_t rv = state_manager_.CheckOperationState(
      FileIOStateManager::OPERATION_EXCLUSIVE, false);
  if (rv != PP_OK)
    return rv;

  int platform_file_flags = 0;
  if (!ppapi::PepperFileOpenFlagsToPlatformFileFlags(open_flags,
                                                     &platform_f
```

Control message route

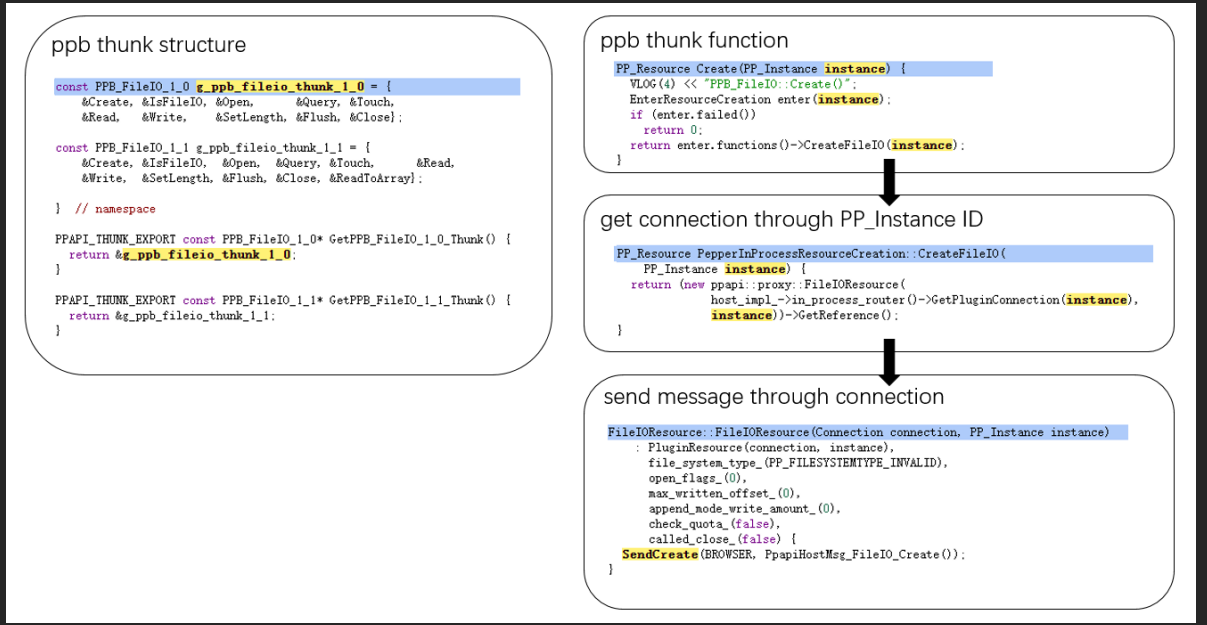
```
// FileIO
IPC_MESSAGE_CONTROL0(PpapiHostMsg_FileIO_Create)
IPC_MESSAGE_CONTROL2(PpapiHostMsg_FileIO_Open,
                     PP_Resource /* file_ref_resource */,
                     int32_t /* open_flags */)
IPC_MESSAGE_CONTROL2(PpapiPluginMsg_FileIO_OpenReply,
                     PP_Resource /* quota_file_system */,
                     int64_t /* file_size */)
IPC_MESSAGE_CONTROL1(PpapiHostMsg_FileIO_Close,
                     ppapi::FileGrowth /* file_growth */)
IPC_MESSAGE_CONTROL2(PpapiHostMsg_FileIO_Touch
```

```
PPAPI_BEGIN_MESSAGE_MAP(PepperFileIOHost, msg)
  PPAPI_DISPATCH_HOST_RESOURCE_CALL(PpapiHostMsg_FileIO_Open, OnHostMsgOpen)
  PPAPI_DISPATCH_HOST_RESOURCE_CALL(PpapiHostMsg_FileIO_Touch, OnHostMsgTouch)
  PPAPI_DISPATCH_HOST_RESOURCE_CALL(PpapiHostMsg_FileIO_SetLength,
                                    OnHostMsgSetLength)
  PPAPI_DISPATCH_HOST_RESOURCE_CALL_0(PpapiHostMsg_FileIO_Flush,
```

Prior Knowledge

Ppapi

- By using the existing connection
- PP_Instance ID <-> Connection
- All PP_Instance ID is stored in a global map in renderer process
- call ppb_thunk function with PP_Instance



The Bug

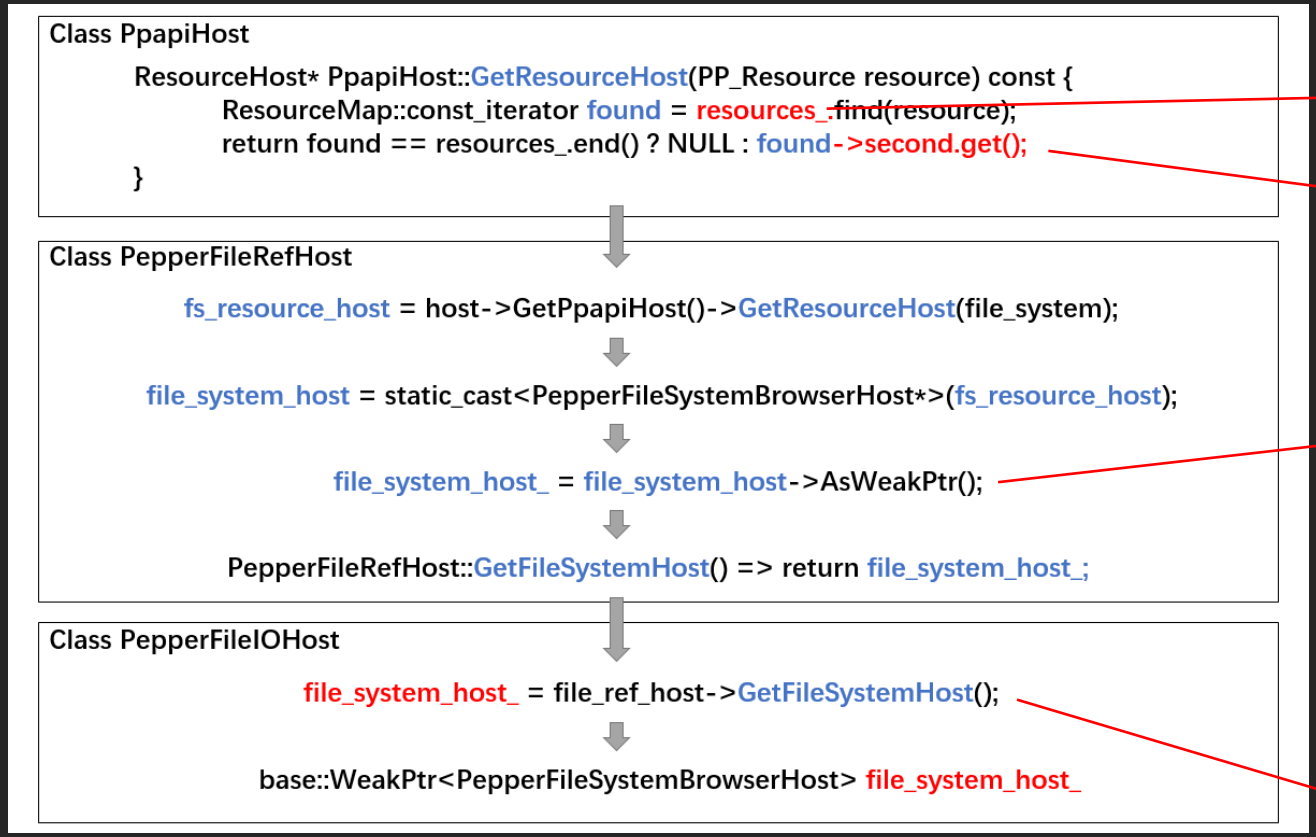
```
std::unique_ptr <T> sptr_x;

raw_x_ = x.get();

sptr_x.reset();

raw_x_->DoSomething();
```

weakptr



The Bug

could be passed in from the renderer-side

```
1  void PpapiHost::OnHostMsgResourceCreated(const proxy::ResourceMessageCallParams& params,
2   …
3   DCHECK(resource_host->pp_resource());
4   resources_[params.pp_resource()] = std::move(resource_host);
5  }
```

```
typedef std::map<PP_Resource, std::unique_ptr<ResourceHost>> ResourceMap;

ResourceMap resources_;
```

base::WeakPtr<PepperFileSystemBrowserHost> file_system_host_

Exploit

```
141  base::WeakPtr<PepperFileSystemBrowserHost> file_system_host_;
142
143  // Valid only for PP_FILESYSTEMTYPE_LOCAL{PERSISTENT,TEMPORARY}.
144  scoped_refptr<storage::FileSystemContext> file_system_context_;
145  storage::FileSystemURL file_system_url_;
146  base::OnceClosure on_close_callback_;
```

历史记录 References

file_system_host_

输入内容以按文件路径过滤

定义 (已显示 1 个)

content/browser/renderer_host/pepper/pepper_file_io_host.h (1)

141: base::WeakPtr<PepperFileSystemBrowserHost> file_system_host_;

参考 (已显示 9 个)

content/browser/renderer_host/pepper/pepper_file_io_host.cc (9)

178: file_system_host_ = file_ref_host->GetFileSystemHost();
250: if (!file_system_host_.get()) {
256: DCHECK(file_system_host_->GetFileSystemOperationRunner());
258: file_system_host_->GetFileSystemOperationRunner()->OpenFile(
272: if (FileOpenForWrite(open_flags_) && file_system_host_->ChecksQuota()) {
274: file_system_host_->OpenQuotaFile(
366: file_system_host_->CloseQuotaFile(this, file_growth);
397: if (open_flags_ != PP_FILEOPENFLAG_READ && file_system_host_->ChecksQuota())
488: quota_file_system = file_system_host_->pp_resource();

```
257
258  file_system_host_->GetFileSystemOperationRunner()->OpenFile(
259      file_system_url_, platform_file_flags,
260      base::BindOnce(&DidOpenFile, AsWeakPtr(), task_runner_,
261                     base::BindOnce(&PepperFileIOHost::DidOpenInternalFile,
262                                    AsWeakPtr(), reply_context)));
263  }
```

```
377
378  OperationID FileSystemOperationRunner::OpenFile(const FileSystemURL& url,
379                                                  int file_flags,
380                                                  OpenFileCallback callback) {
381    base::File::Error error = base::File::FILE_OK;
382    std::unique_ptr<FileSystemOperation> operation = base::WrapUnique(
383        file_system_context_->CreateFileSystemOperation(url, &error));
384    FileSystemOperation* operation_raw = operation.get();
```

```
534
535  FileSystemOperation* FileSystemContext::CreateFileSystemOperation(
536      const FileSystemURL& url,
537      base::File::Error* error_code) {
538    if (!url.is_valid()) {
539      if (error_code)
540        *error_code = base::File::FILE_ERROR_INVALID_URL;
541      return nullptr;
542    }
543
544    FileSystemBackend* backend = GetFileSystemBackend(url.type());
545    if (!backend) {
546      if (error_code)
547        *error_code = base::File::FILE_ERROR_FAILED;
548      return nullptr;
549    }
550
551    base::File::Error fs_error = base::File::FILE_OK;
552    FileSystemOperation* operation =
553        backend->CreateFileSystemOperation(url, this, &fs_error);
554
```

```
98   virtual FileSystemOperation* CreateFileSystemOperation(
99       const FileSystemURL& url,
100      FileSystemContext* context,
101      base::File::Error* error_code) const = 0;
```

Exploit

Need to construct the structure to meet the constraints on this path and hijack the control flow.

No inter-process randomization on Windows.

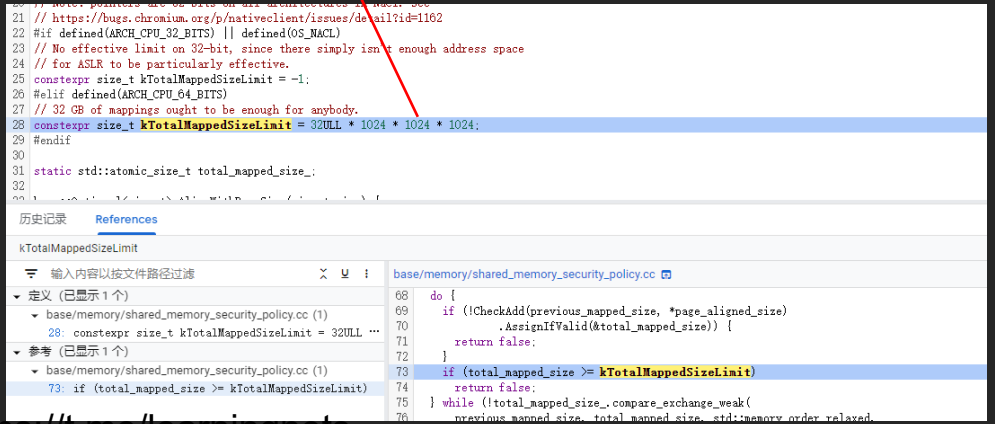
Only need to leak the heap address

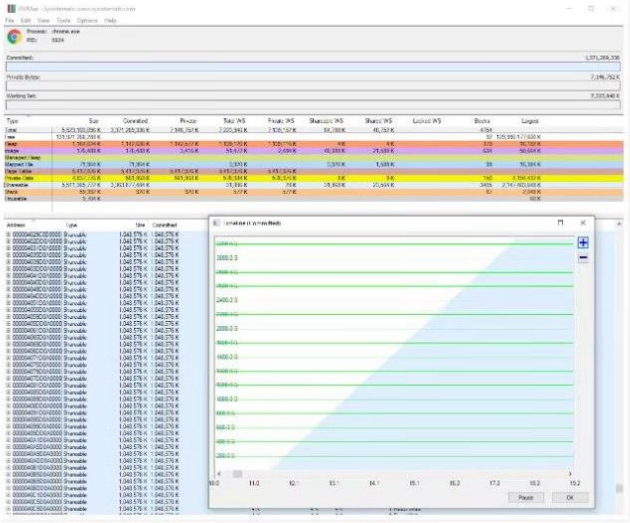
Exploit

Leak the heap address

Use SharedBuffers as Mark Brand[*]?

32GB limited!





As we can see in the VMMap screenshot above - this is both effective and quick! The first test performed a 16-terabyte spray, which got a bit laggy, but in the real-world about 3.5-terabytes appears sufficient to get a reliable, predictable address. Finally, a chance to cite SkyLined's exploit for MS04-040 in a modern 64-bit Chrome exploit!

[*] https://googleprojectzero.blogspot.com/2019/04/virtually-unlimited-memory-escaping.html

Exploit

Reply 4 bytes to renderer process

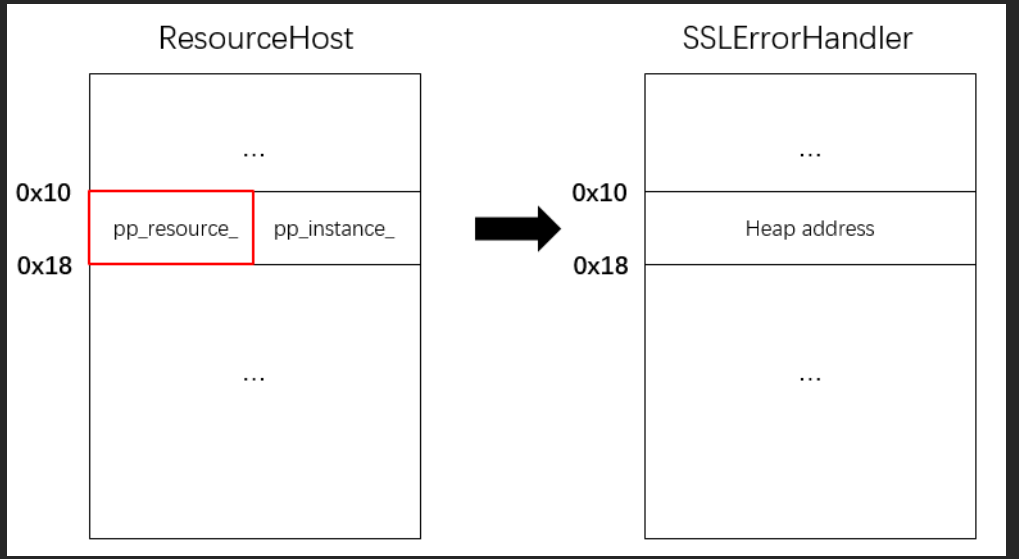
```
1  void PepperFileIOHost::SendFileOpenReply(
2   …
3   if (pp_error == PP_OK) {
4    state_manager_.SetOpenSucceed();
5    // A non-zero resource id signals the plugin side to check quota.
6    if (check_quota_)
7      quota_file_system = file_system_host_->pp_resource();
8   }
9   reply_context.params.set_result(pp_error);
10  host()->SendReply(
11      reply_context,
12      PpapiPluginMsg_FileIO_OpenReply(quota_file_system, max_written_offset_));
13   state_manager_.SetOperationFinished();
14 }
```

```
0:015> dt chrome!content::PepperFileSystemHost
   +0x000 __VFN_table : Ptr64
   +0x008 host_            : Ptr64 ppapi::host::PpapiHost
   +0x010 pp_instance_     : Int4B
   +0x014 pp_resource_     : Int4B
   +0x018 message_filters_ : std::__1::vector<scoped_refptr<ppapi::
   +0x038 weak_reference_owner_ : base::internal::WeakReferenceOwne
   +0x040 renderer_ppapi_host_ : Ptr64 content::RendererPpapiHost
   +0x048 reply_context_   : ppapi::host::ReplyMessageContext
   +0x078 type_            : <unnamed-tag>
   +0x07c opened_          : Bool
   +0x080 root_url_        : GURL
   +0x0f8 called_open_     : Bool
   +0x100 file_system_manager_ : mojo::Remote<blink::mojom::FileSys
```

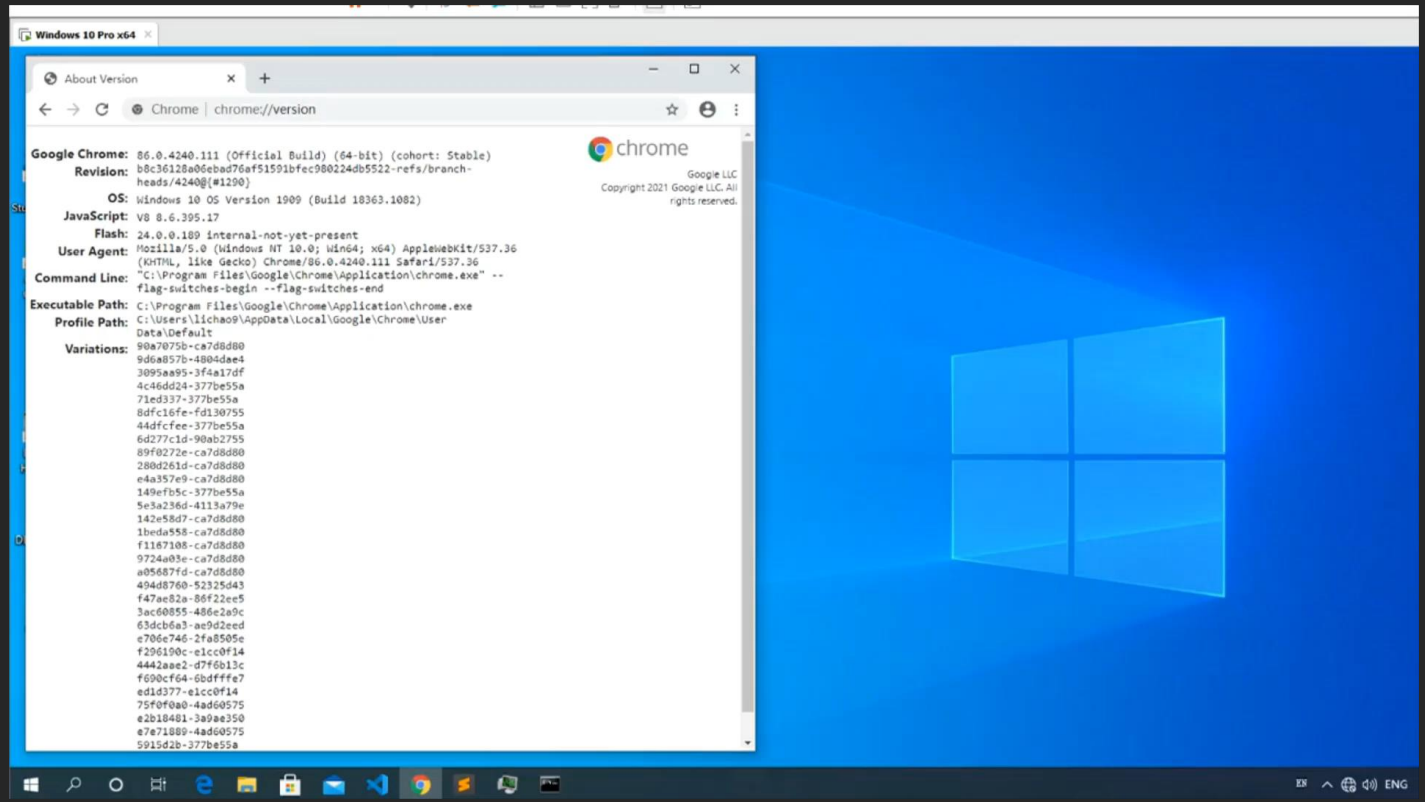
Exploit

```
0:015> dt chrome!content::PepperFileSystemHost
   +0x000 __VFN_table : Ptr64
   +0x008 host_              : Ptr64 ppapi::host::PpapiHost
   +0x010 pp_instance_       : Int4B
   +0x014 pp_resource_       : Int4B
   +0x018 message_filters_   : std::__1::vector<scoped_refptr<ppapi::
   +0x038 weak_reference_owner_ : base::internal::WeakReferenceOwne
   +0x040 renderer_ppapi_host_ : Ptr64 content::RendererPpapiHost
   +0x048 reply_context_     : ppapi::host::ReplyMessageContext
   +0x078 type_              : <unnamed-tag>
   +0x07c opened_            : Bool
   +0x080 root_url_          : GURL
   +0x0f8 called_open_       : Bool
   +0x100 file_system_manager_ : mojo::Remote<blink::mojom::FileSys
```

```
0:015> dt chrome!content::SSLErrorHandler
   +0x000 __VFN_table : Ptr64
   +0x008 delegate_          : base::WeakPtr<content::SSLErrorHandler::Delegate>
   +0x010 request_url_       : GURL
   +0x090 is_main_frame_request_ : Bool
   +0x098 ssl_info_          : net::SSLInfo
   +0x120 cert_error_        : Int4B
   +0x124 fatal_             : Bool
   +0x128 web_contents_      : Ptr64 content::WebContents
0:015> dt chrome!base::WeakPtr<content::SSLErrorHandler::Delegate>
   +0x000 ref_               : base::internal::WeakReference
   +0x008 ptr_               : Uint8B
```



Demo Video



CONCLUSION

- Some background
- Some bug and pattern, how to find bug variants
  - RenderFrameHost lifetime issues
  - Error Return pattern
  - WeakPtr Optimization
- The exploit

THANKS!