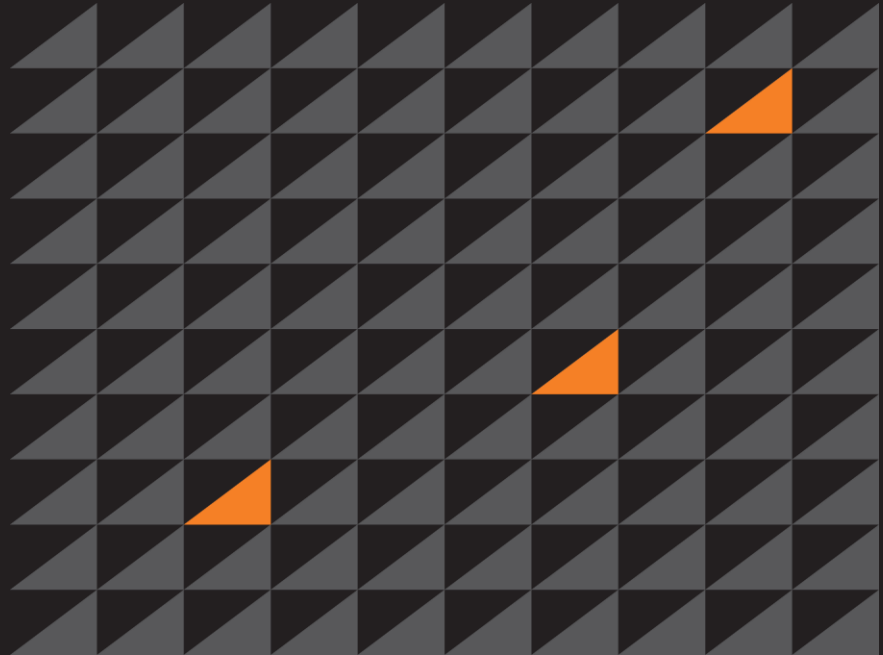




Poor Man's Static Analysis

BSides London 2014

Jon Butler, MWR InfoSecurity



whoami



- **Head of Research @ MWR**
 - Bugs <3
- **Troll: @securitea**



Why?

- **Current tool set**
 - Text-based pattern matching (grep)
 - Manually review each match
 - Fully automated code scanner
 - List of false positives
- **No middle ground?**



This talk

- **Practical, not academic**
- **Assisted, not automatic**
- **Not a comprehensive overview**

- **A story...**



Once upon a time...

- **Long-term research**
 - Google Chrome
- **UAF**
 - 80% of reported bugs [citation needed]
 - Usually fuzzed
- **New approach == new bugs?**



If I were a UAF...

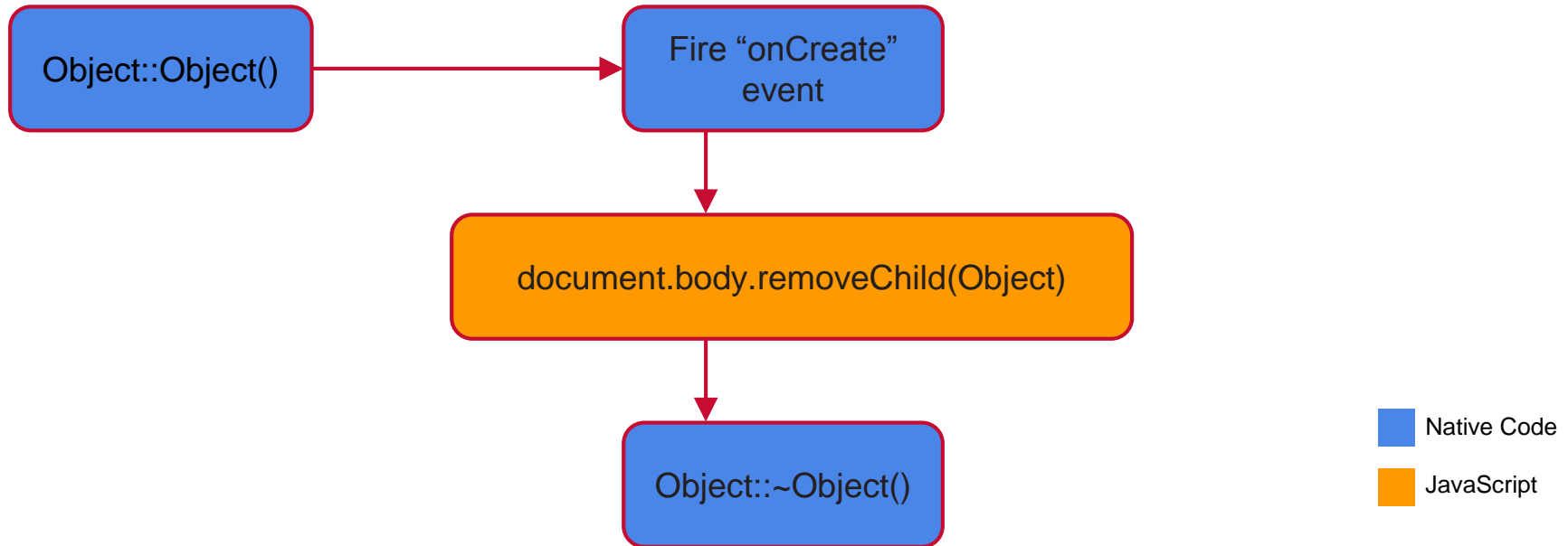
- **Where would I be hiding?**
 - Likely, in asynchronous code
 - Web Audio?
 - IDB?
 - DOM Events?

DOM Events (Simplified)

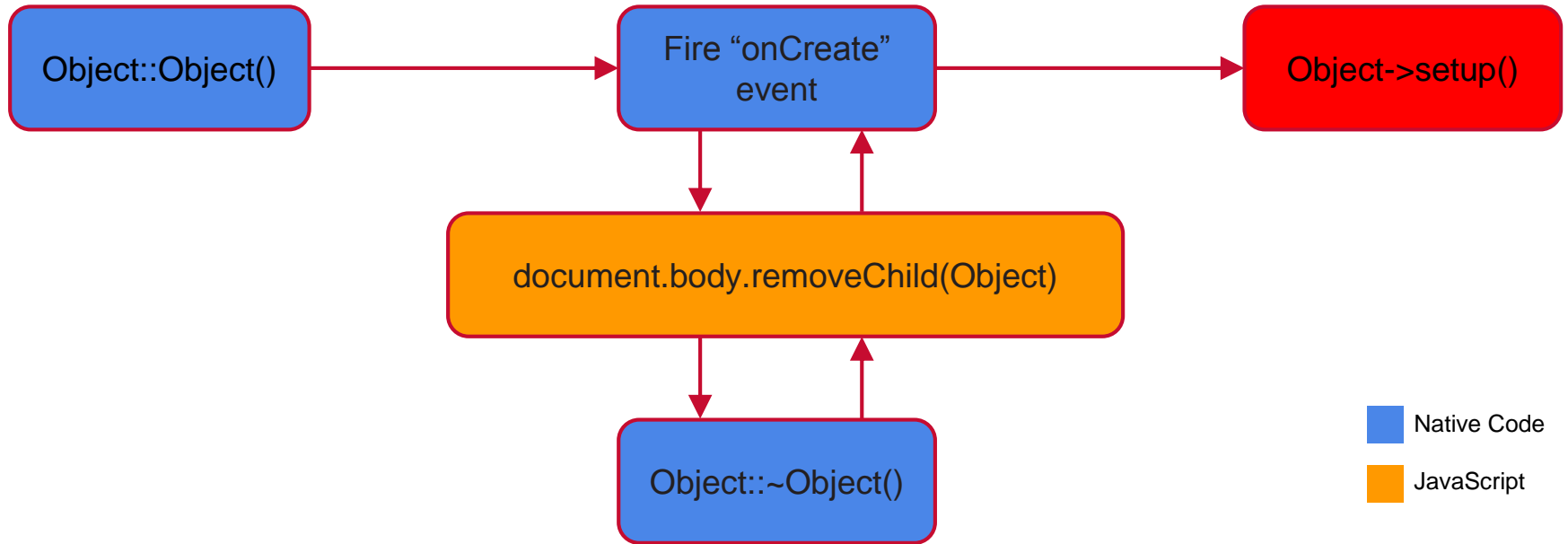


■ Native Code
■ JavaScript

DOM Events (Simplified)



DOM Events (Simplified)





Our “Pattern”

- **A scope with a raw pointer to an object...**
- **Followed by a function call that dispatches an event...**
- **Which can free all remaining references to the object...**
- **Followed by a use of the freed object**



Our “Pattern”

- **A scope with a raw pointer to an object...**
- **Followed by a function call that dispatches an event...**
- **Which can free all remaining references to the object...**
- **Followed by a use of the freed object**



Easy, right?

- **Which functions trigger events?**
 - Node::dispatchEvent?
 - Duh...
 - How about InputType::applyStep?
 - Doesn't directly call dispatchEvent...



Easy, right?

- **Sadly not the first to think of this...**
- **Protections against the obvious**
 - Smart pointers
 - RefPtr et al.
 - Event queue objects
 - Prevent events dispatching while queue object is in scope



Approach

- **Clearly, grep is out of its league**
- **AFAIK, no suitable tool exists**
- **We need a tool that**
 - Knows about C++ (and its complexities)
 - Knows about the control flow in the target
 - Can provide this info in a flexible way



Solution

- **Chrome is compiled with Clang**
 - Extendable
 - **Has** to know about C++
 - **Has** to know about call flow
- **No immediate way to get info out**
- **Three interfaces for interaction**



Clang Interfaces

Interface	Pros	Cons
Clang Plugins	<ul style="list-style-type: none">- Integrated into build process- Perfect for compile-time code checking	<ul style="list-style-type: none">- Runs on each build, not standalone
libclang	<ul style="list-style-type: none">- Written in C, Python bindings (swig)- High-level, easy to get started	<ul style="list-style-type: none">- Not enough functionality exposed- Unstable API
libtooling	<ul style="list-style-type: none">- All of Clang's functionality exposed- Flexible- Documented (kinda)	<ul style="list-style-type: none">- Complex API (C++)- Single-threaded

Enter: CallsDeep

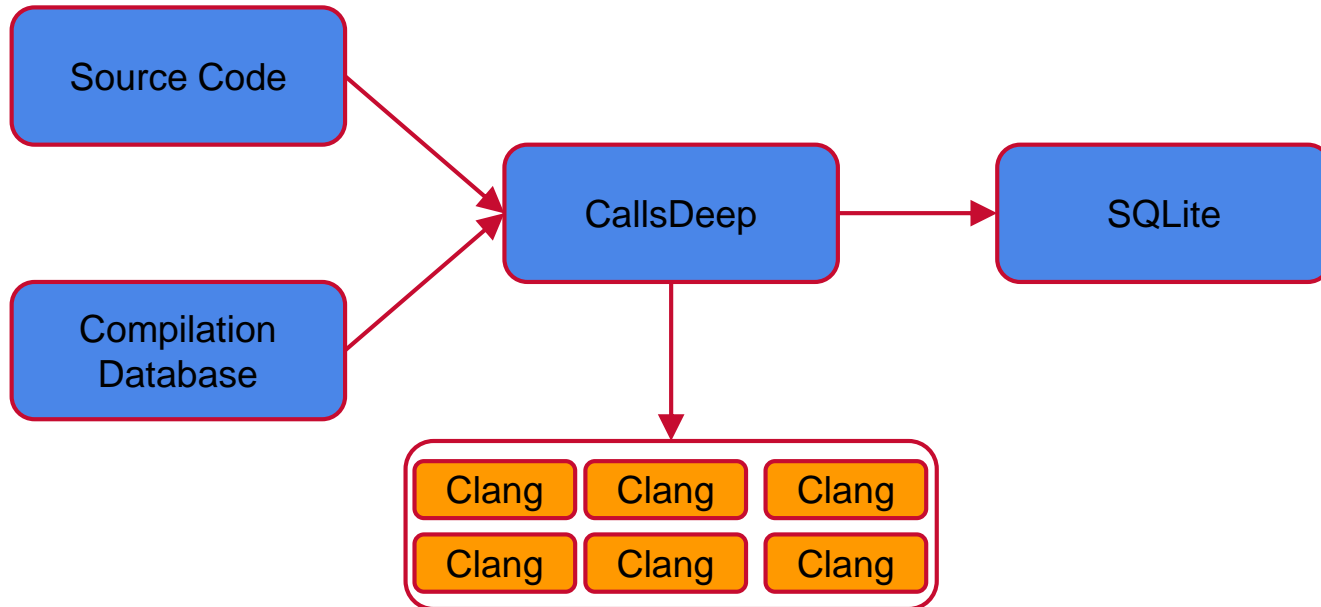
- **Clang tool**
 - Extends Clang's RecursiveASTVisitor class
 - Function definitions
 - Function calls
 - Variable declaration
- **Python core**
 - 1 x Clang tool per CPU
 - Collates results
 - Outputs to SQLite DB



Enter: CallsDeep

- **Various presentation formats**
 - SQL
 - GraphML
 - Text
 - Web (experimental)

Enter: CallsDeep





Compilation Database?

- **Build arguments for each file**
- **ninja**
 - -t compdb cxx
- **cmake**
 - -DCMAKE_EXPORT_COMPILE_COMMANDS=ON
- **BEAR**



Core Run

```
jon@brute:~/research/callsdeep$ time callsdeep.py output.db
Parsing file set, 5641 items
Found 70886 functions, 264116 calls, and 137722 variables
Preparing parameters
Saving to database
Adding indexes

real    77m58.207s
```



Raw SQL

```
sqlite> select qualname from functions where qualname like '%dispatchEvent%';
WebCore::BlurEventDispatchMediator::dispatchEvent(class WebCore::EventDispatcher *)
const
WebCore::DOMWindow::dispatchEvent(WTF::PassRefPtr<Event>, PassRefPtr<class
WebCore::EventTarget>)
WebCore::DOMWindowEventQueue::dispatchEvent(WTF::PassRefPtr<Event>)
WebCore::DataListIndicatorElement::preDispatchEventHandler(class WebCore::Event *)
...
```



Raw SQL

```
sqlite> select callername from calls where calleename like  
'%WebCore::DOMWindow::dispatchEvent%';  
WebCore::DOMWindow::dispatchLoadEvent()  
WebCore::Document::dispatchBeforeUnloadEvent(class WebCore::Chrome &, _Bool &)  
WebCore::ScriptedAnimationController::dispatchEvents()  
WebCore::DOMWindowEventQueue::dispatchEvent(WTF::PassRefPtr<Event>)  
WebCore::Document::dispatchUnloadEvents()
```



What next?

- **Graph of calls to event dispatches**
- **Pick a graphing library...**
 - Good *nix-based viewer / editor
 - Python library support
- **yEd, GraphML**



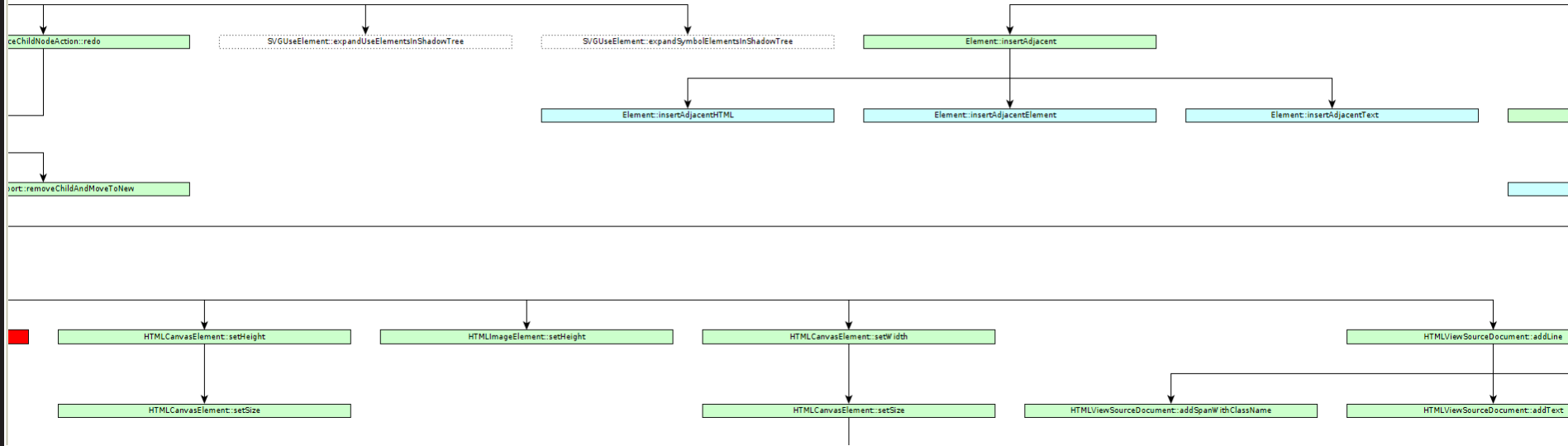
Graphing

- **Specify start point**
- **Find calls**
- **Get calling function**
- **Repeat**
 - Breadth-first
- **Python networkx**

Graphing



Graphing



“Sniper”

- **More information**
- **Can we get more specific?**
 - Variable declaration
 - Event dispatch
 - Use?



“Sniper”

```
Processing calls to 'WebCore::Node::dispatchSubtreeModifiedEvent()'...
```

```
Candidate @ third_party/WebKit/Source/core/html/forms/TextFieldInputType.cpp:359
```

```
Parent Function: WebCore::TextFieldInputType::listAttributeTargetChanged()
```

```
Call: WebCore::ContainerNode::appendChild(PassRefPtr<class WebCore::Node>, class  
      WebCore::ExceptionState &) on line 359 fires event
```

```
Use: picker on line 361
```

```
Candidate @ third_party/WebKit/Source/core/html/forms/SearchInputType.cpp:98
```

```
Parent Function: WebCore::SearchInputType::createShadowSubtree()
```

```
Call: WebCore::ContainerNode::insertBefore(PassRefPtr<class WebCore::Node>, class  
      WebCore::Node *, class WebCore::ExceptionState &) on line 98 fires event
```

```
Use: container on line 99
```

```
Use: viewport on line 99
```

```
...
```



Improvements

- **map vs imap_unordered**
- **Override resolution**
- **Actual software development**
 - SQL indexes
- **Blacklisting**
 - Mocks / tests
 - Accessibility & known non-defaults

Wider Use

- **Has some restrictions**
 - Must use Clang
 - Requires compdb
 - Depends on build system
 - Application specific



Wider Use

- **Works with Clang**
 - Number of FOSS
 - Others?
 - Code changes?
- **IOS?**
 - XCode integration in progress...



Thanks for Listening!

Questions?