

Source Code Analysis for Security through LLVM

Lu Zhao

HP Fortify

lu.zhao@hp.com

Static Code Analyzer for Security

```

1#import "HtmlViewController.h"
2
3@implementation HtmlViewController
4
5@synthesize webView, content = _content;
6
7-(void)viewDidLoad {
8    NSString *resourcePath = [[NSBundle mainBundle] resourcePath];
9    NSURL *baseURL = [[NSURL alloc] initWithURLWithPath:
10    resourcePath isDirectory:YES];
11
12    if (_content) {
13        // Render the existing content in the web view.
14        [self.webView loadHTMLString:_content baseURL:baseURL];
15    } else {
16        // Display the "About iGoat" splash screen as a default.
17        NSError *error;
18        NSString *filePath = [[NSBundle mainBundle]
19        pathForResource:@"splash.html" ofType: nil];
20        NSString *fileContents = [[NSString alloc]
21        initWithContentsOfFile:filePath encoding:
22        NSUTF8StringEncoding error:&error];
23        NSString *version = [[NSBundle mainBundle] infoDictionary
24        objectForKey:@"CFBundleShortVersionString"];
25
26        [self.webView loadHTMLString:[NSString stringWithFormat:
27            fileContents, version] baseURL:baseURL];
28    }
29
30    [super viewDidLoad];
31}
32
33-(void)setContent:(NSString *)newContent {
34    // Format the content as HTML if necessary.
35    _content = [self formatAsHTML:newContent];
36
37    // Render the content in the web view.
38    // TODO: DRY this up (see above).
39    NSString *resourcePath = [[NSBundle mainBundle] resourcePath];
40    NSURL *baseURL = [[NSURL alloc] initWithURLWithPath:
41    resourcePath isDirectory:YES];

```

Fortify
SCA



The screenshot shows the Fortify SCA interface. On the left, there is a code editor window showing the same `HtmlViewController.m` code as the previous image. To the right of the code editor is a large window titled "File Edit Tools Options Help" and "Summary | Audit Guide | Scan | Reports". The main pane displays a "Critical (13) Hidden (0) Removed (0) Suppressed (0)" section. Under this, there is a tree view of security issues: "Cross-Site Scripting: Persistent - [0 / 3]" (with three sub-items), "JSON Injection - [0 / 1]", "Privacy Violation - [0 / 0]", and "SQL Injection - [0 / 1]". Below the tree view is a "Analysis Evidence" section with several items, each with a small icon and some text. At the bottom of the main pane, there is a summary of the issue: "Issue: HtmlViewController.m:21 (Cross-Site Scripting: Persis". On the right side of the interface, there is a vertical sidebar with "Project Summary" and "Audit Workbench" tabs. The "Audit Workbench" tab is active, showing a detailed view of the selected issue. This view includes fields for "User:" and "Ana...", a "Edit..." button, and a large text area containing a detailed description of the issue: "The method of viewDidLoad() in HtmlViewController.m sends unvalidated data to a web browser on line 21, which can result in the browser executing malicious code." There are also "More Information..." and "Recommendations..." links at the bottom of this panel.

File Edit Tools Options Help

Audit Workbench (on Iupc3)

Summary | Audit Guide | Scan | Reports

AUDIT WORKBENCH FORTIFY

Filter Set: Quick View ▾ My Issues

Critical (13) Hidden (0) Removed (0) Suppressed (0)

Group By: Category

13 54 ... 67

Cross-Site Scripting: Persistent - [0 / 3]

- DetailViewController.m:51 (Cross-Site Scripting: Persistent)
- HtmlViewController.m:21 (Cross-Site Scripting: Persistent)
- HtmlViewController.m:35 (Cross-Site Scripting: Persistent)

JSON Injection - [0 / 1]

Privacy Violation - [0 / 8]

SQL Injection - [0 / 1]

Advanced...

Analysis Evidence

- HtmlViewController.m:18 - initWithContentsOfFile:encoding:error:(return)
- HtmlViewController.m:18 - Assignment to fileContents
- HtmlViewController.m:21 - stringWithFormat:(0 : return)
- HtmlViewController.m:21 - loadHTMLString:baseURL:(0)

Rule ID: 5F9C56F0-C87C-4390-A56E-CD8BA1885E64

Taint Flags: FILE_SYSTEM, XML, XSS

Direct Function Call: UIWebView::loadHTMLString:baseURL:()

Project Summary | HtmlViewController.m

```
1#import "HtmlViewController.h"
2
3@implementation HtmlViewController
4
5@synthesize webView, content = _content;
6
7- (void)viewDidLoad {
8    NSString *resourcePath = [[NSBundle mainBundle] resourcePath];
9    NSURL *baseUrl = [[NSURL alloc] initFileURLWithPath:resourcePath is|
10
11    if (_content) {
12        // Render the existing content in the web view.
13        [self.webView loadHTMLString:_content baseURL:baseUrl];
14    } else {
15        // Display the "About iGoat" splash screen as a default.
16        NSError *error;
17        NSString *filePath = [[NSBundle mainBundle] pathForResource:@"s|"
18        NSString *fileContents = [[NSString alloc] initWithContentsOfFi|
19        NSString *version = [[[NSBundle mainBundle] infoDictionary] obj|
20
21        [self.webView loadHTMLString:[NSString stringWithFormat:fileCon|
22    }
23
24    [super viewDidLoad];
25}
26
27- (void)setContent:(NSString *)newContent {
28    // Format the content as HTML if necessary.
29    _content = [self formatAsHTML:newContent];
30}
```

Summary | Details | Recommendations | History | Diagram | Screenshots | Filters

Issue: HtmlViewController.m:21 (Cross-Site Scripting: Persis ▾

User:

Ana...

Edit...

Cross-Site Scripting: Persistent (Input Validation and Representation, Data Flow)

The method viewDidLoad() in HtmlViewController.m sends unvalidated data to a web browser on line 21, which can result in the browser executing malicious code.

Click to append comment (Ctrl +Enter to save)

More Information... Recommendations...

File Edit Tools Options Help

Audit Workbench (on Iupc3)

[Summary](#) | [Audit Guide](#) | [Scan](#) | [Reports](#)

AUDIT WORKBENCH FORTIFY

Filter Set: Quick View ▾

My Issues

Critical (13) Hidden (0) Removed (0) Suppressed (0)

Group By: Category

Cross-Site Scripting: Persistent - [0 / 3]

- DetailViewController.m:51 (Cross-Site Scripting: Persistent)
- HtmlViewController.m:21 (Cross-Site Scripting: Persistent)
- HtmlViewController.m:35 (Cross-Site Scripting: Persistent)

JSON Injection - [0 / 1]

Privacy Violation - [0 / 8]

SQL Injection - [0 / 1]

Project Summary HtmlViewController.m

```
1#import "HtmlViewController.h"
2
3@implementation HtmlViewController
4
5@synthesize webView, content = _content;
6
7- (void)viewDidLoad {
8    NSString *resourcePath = [[NSBundle mainBundle] resourcePath];
9    NSURL *baseURL = [[NSURL alloc] initFileURLWithPath:resourcePath is|
10
11    if (_content) {
12        // Render the existing content in the web view.
13        [self.webView loadHTMLString:_content baseURL:baseURL];
14    } else {
15        // Display the "About iGoat" splash screen as a default.
16        NSError *error;
17        NSString *filePath = [[NSBundle mainBundle] pathForResource:@"s|"
18        NSString *fileContents = [[NSString alloc] initWithContentsOfFile|
19        NSString *version = [[[NSBundle mainBundle] infoDictionary] obj|
20
21        [self.webView loadHTMLString:[NSString stringWithFormat:fileCon|
22    }
23
24    [super viewDidLoad];
25}
26
27- (void)setContent:(NSString *)newContent {
28    // Format the content as HTML if necessary.
29    _content = [self formatAsHTML:newContent];
30}
```

Analysis Evidence

- HtmlViewController.m:18 - initWithContentsOfFile:encoding:error:(return)
- HtmlViewController.m:18 - Assignment to fileContents
- HtmlViewController.m:21 - stringWithFormat:(0 : return)
- HtmlViewController.m:21 - loadHTMLString:baseURL:(0)

Advanced...

Rule ID: 5F9C56F0-C87C-4390-A56E-CD8BA1885E64

Taint Flags: FILE_SYSTEM, XML, XSS

Direct Function Call: UIWebView::loadHTMLString:baseURL:()

Issue: HtmlViewController.m:21 (Cross-Site Scripting: Persis ▾

User:

Ana...

Edit...

Cross-Site Scripting: Persistent (Input Validation and Representation, Data Flow)

The method `viewDidLoad()` in `HtmlViewController.m` sends unvalidated data to a web browser on line 21, which can result in the browser executing malicious code.

Click to append comment (Ctrl +Enter to save)

More Information... Recommendations...

*owasp-igoat-2.1.sourceanalyzer - Audit Workbench (on Iupc3)

File Edit Tools Options Help

Summary | Audit Guide | Scan | Reports

Filter Set: Quick View ▾

My Issues

Critical (13) Hidden (0) Removed (0) Suppressed (0)

Group By: Category

13 54 ... 67

Cross-Site Scripting: Persistent - [0 / 3]

- DetailViewController.m:51 (Cross-Site Scripting: Persistent)
- HtmlViewController.m:21 (Cross-Site Scripting: Persistent)
- HtmlViewController.m:35 (Cross-Site Scripting: Persistent)

JSON Injection - [0 / 1]

Privacy Violation - [0 / 8]

SQL Injection - [0 / 1]

Analysis Evidence

Advanced...

- HtmlViewController.m:18 - initWithContentsOfFile:encoding:error:(return)
- HtmlViewController.m:18 - Assignment to fileContents
- HtmlViewController.m:21 - stringWithFormat:(0 : return)
- HtmlViewController.m:21 - loadHTMLString:baseURL:(0)

Rule ID: 5F9C56F0-C87C-4390-A56E-CD8BA1885E64

Taint Flags: FILE_SYSTEM, XML, XSS

Direct Function Call: UIWebView::loadHTMLString:baseURL:()

Project Summary | HtmlViewController.m

```
1#import "HtmlViewController.h"
2
3@implementation HtmlViewController
4
5@synthesize webView, content = _content;
6
7- (void)viewDidLoad {
8    NSString *resourcePath = [[NSBundle mainBundle] resourcePath];
9    NSURL *baseUrl = [[NSURL alloc] initFileURLWithPath:resourcePath is];
10
11    if (_content) {
12        // Render the existing content in the web view.
13        [self.webView loadHTMLString:_content baseURL:baseUrl];
14    } else {
15        // Display the "About iGoat" splash screen as a default.
16        NSError *error;
17        NSString *filePath = [[NSBundle mainBundle] pathForResource:@"s";
18        NSString *fileContents = [[NSString alloc] initWithContentsOfFile:filePath];
19        NSString *version = [[[NSBundle mainBundle] infoDictionary] obje;
20
21        [self.webView loadHTMLString:[NSString stringWithFormat:@"%@", fileContents];
22    }
23
24    [super viewDidLoad];
25}
26
27- (void)setContent:(NSString *)newContent {
28    // Format the content as HTML if necessary.
29    _content = [self formatAsHTML:newContent];
30}
```

Issue: HtmlViewController.m:21 (Cross-Site Scripting: Persistent)

User:

Ana...

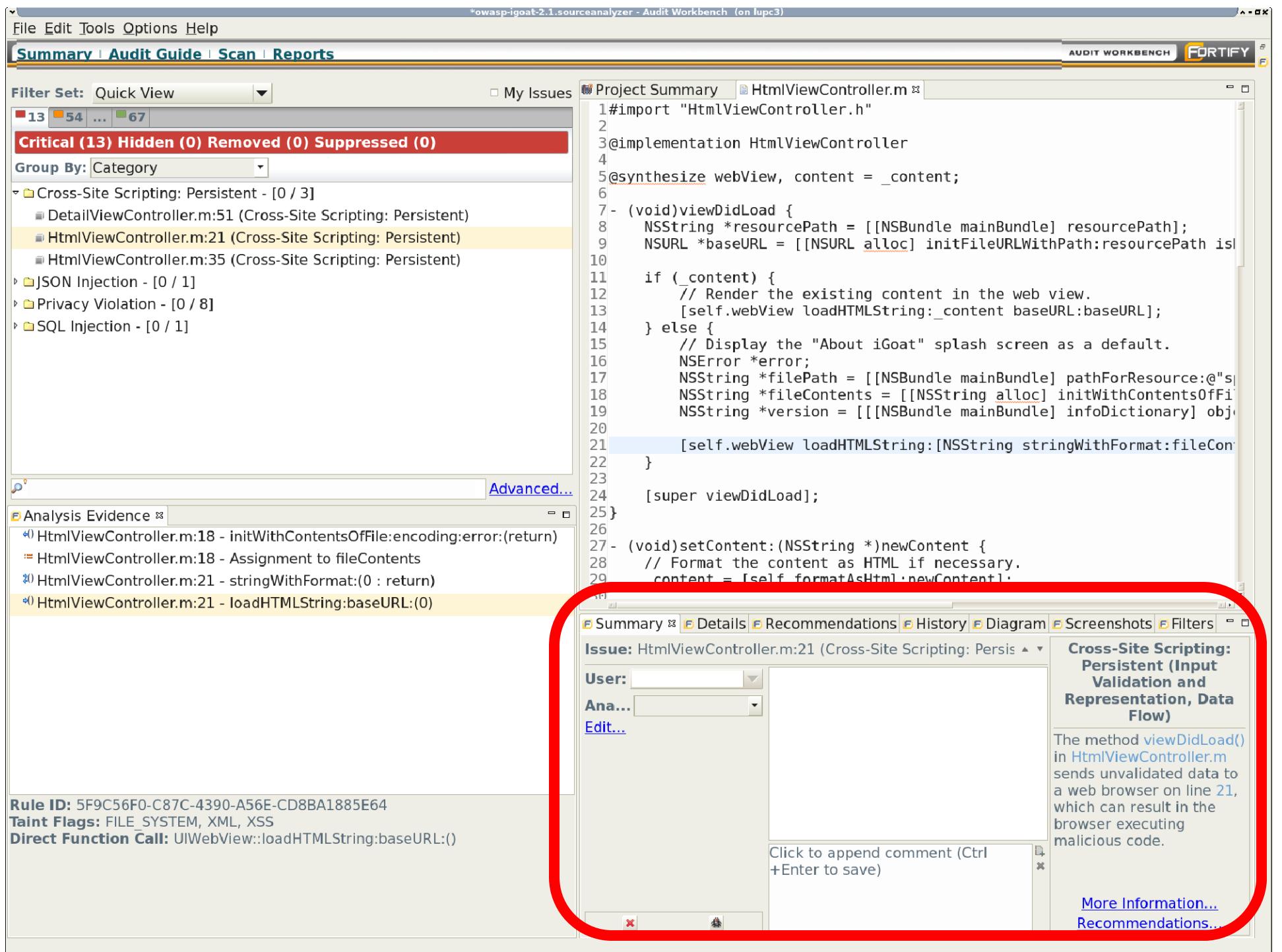
Edit...

Cross-Site Scripting: Persistent (Input Validation and Representation, Data Flow)

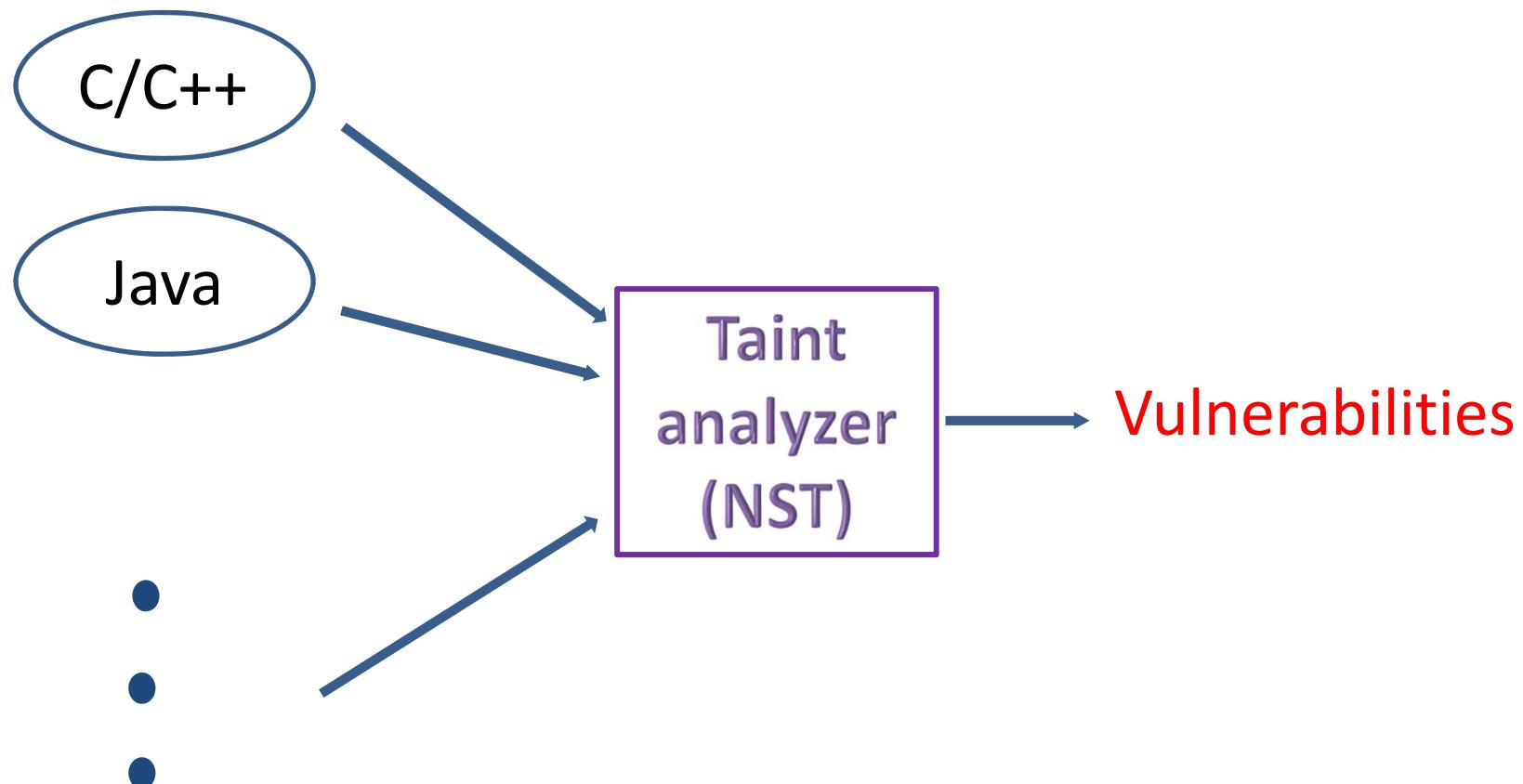
The method viewDidLoad() in HtmlViewController.m sends unvalidated data to a web browser on line 21, which can result in the browser executing malicious code.

Click to append comment (Ctrl +Enter to save)

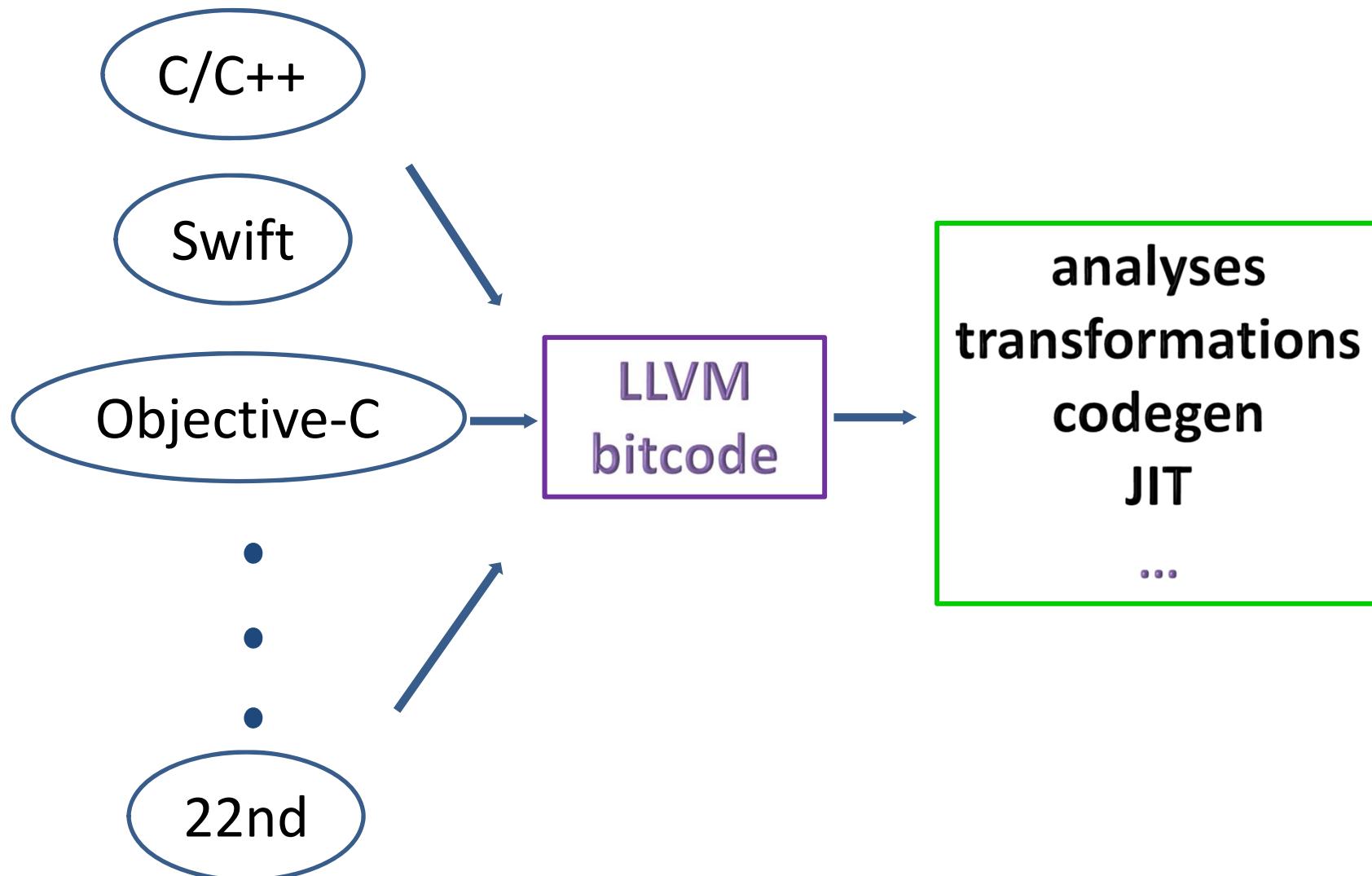
More Information... Recommendations...



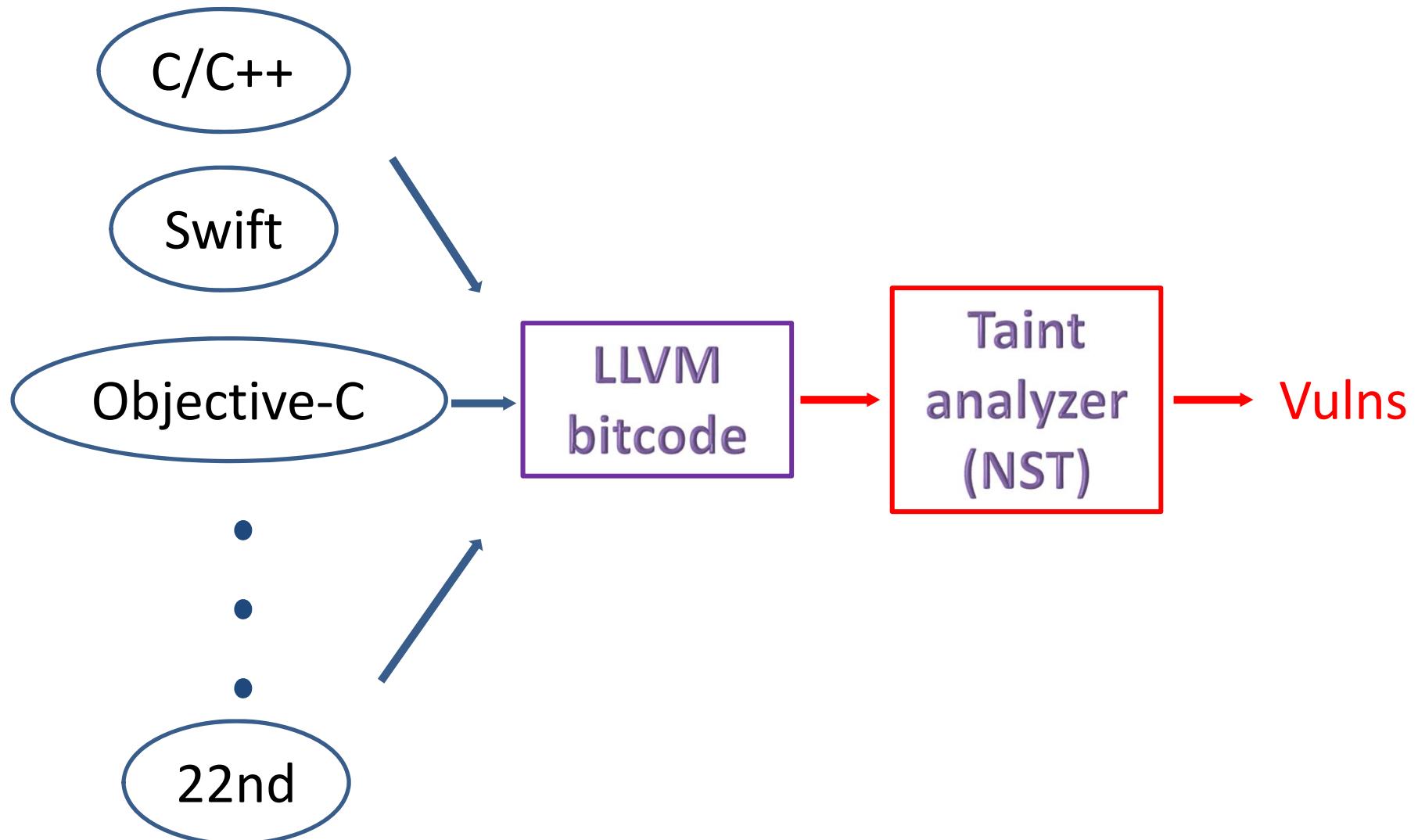
Static Code Analyzer for Security (HP Fortify SCA)



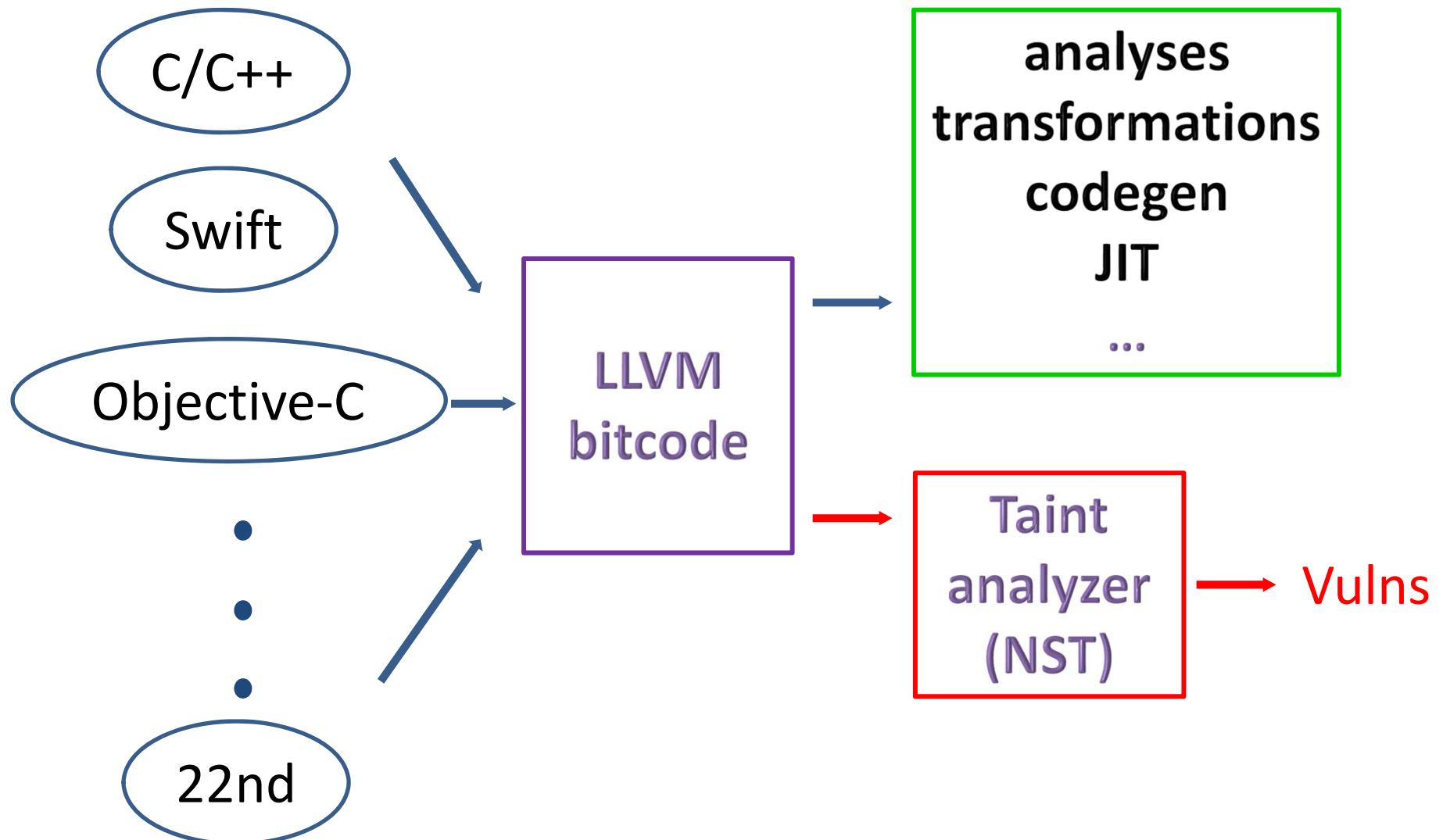
LLVM Language-independent Services



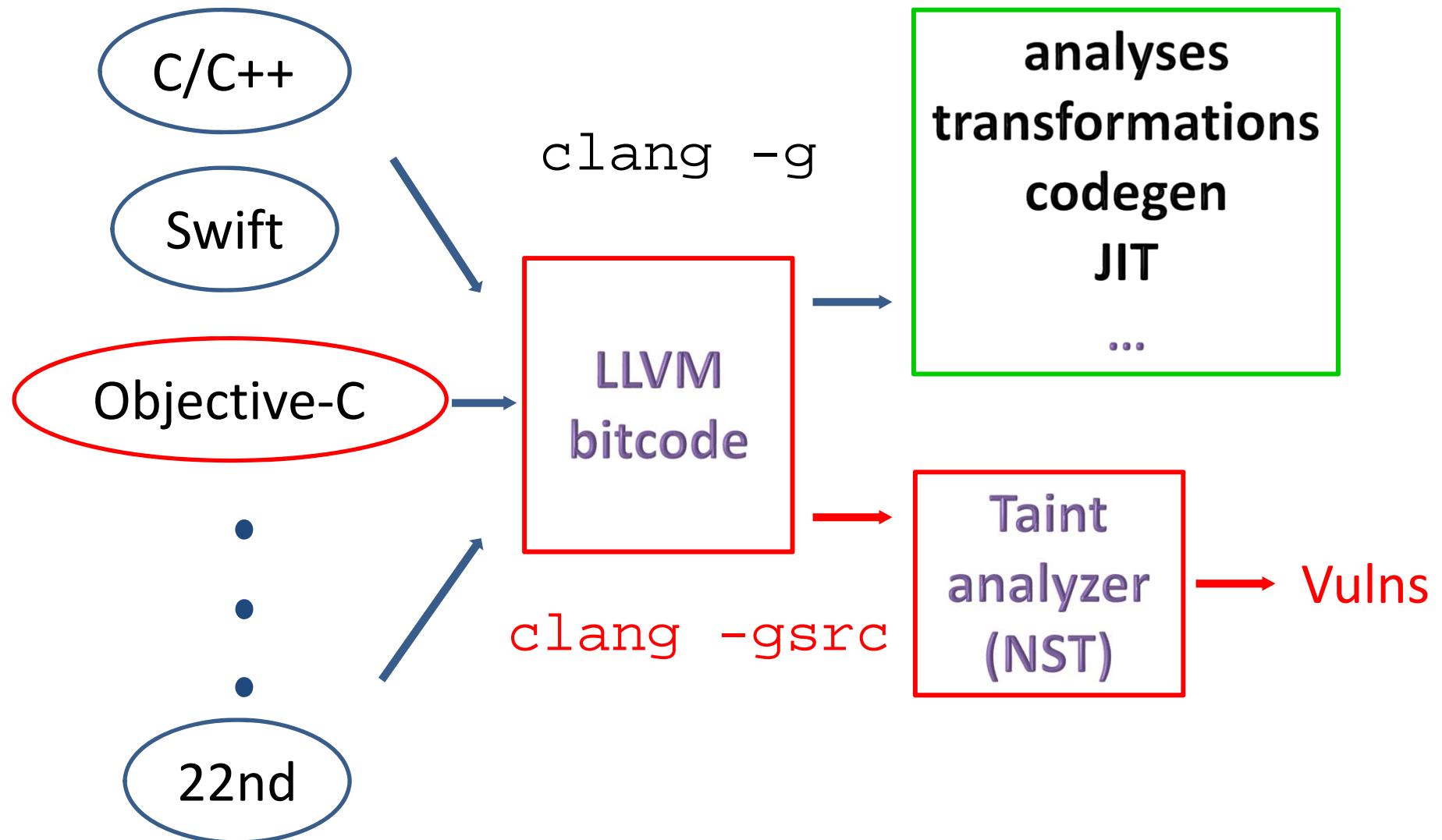
Bitcode for Source Analysis?



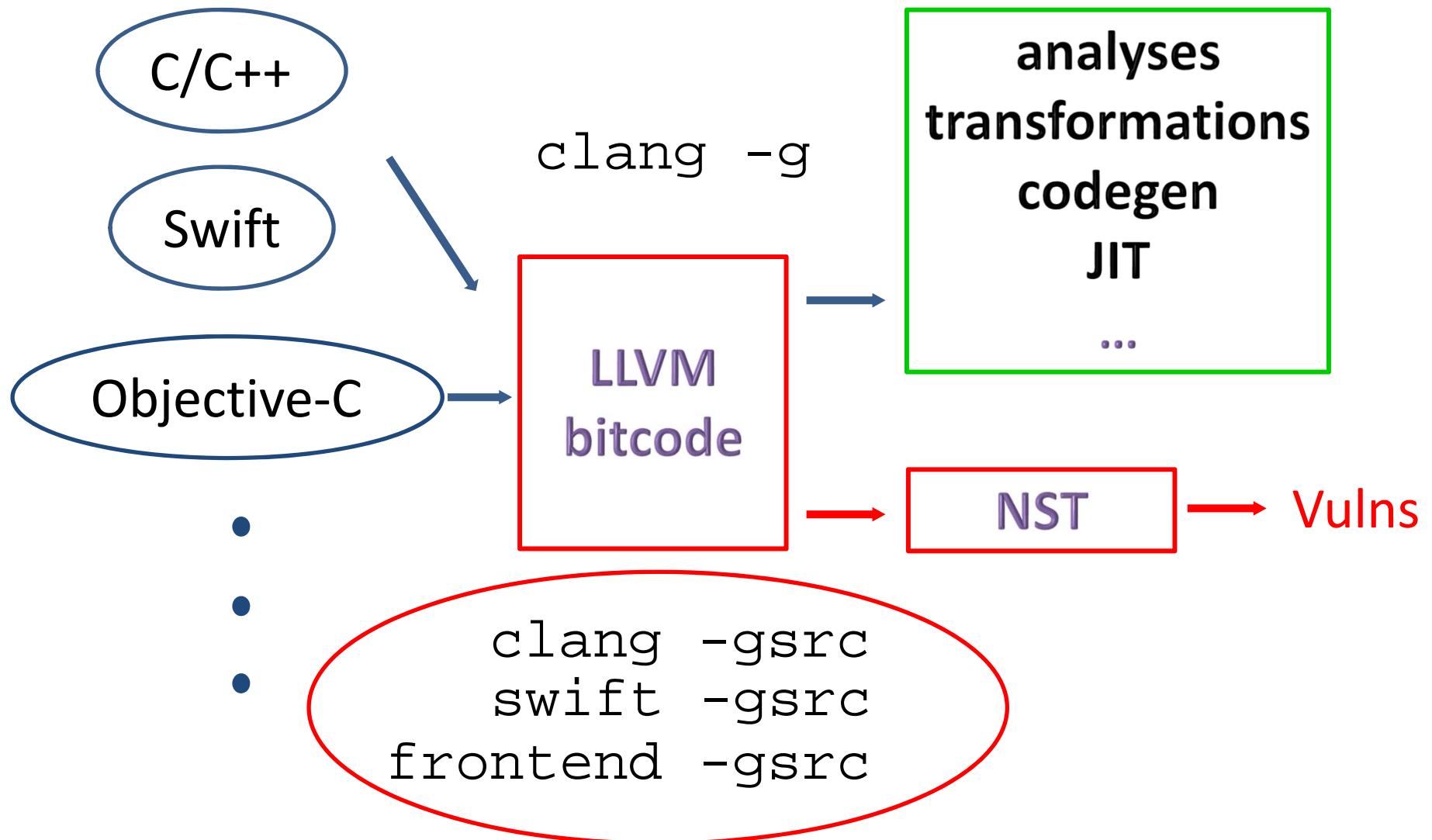
Bitcode for Source Analysis?



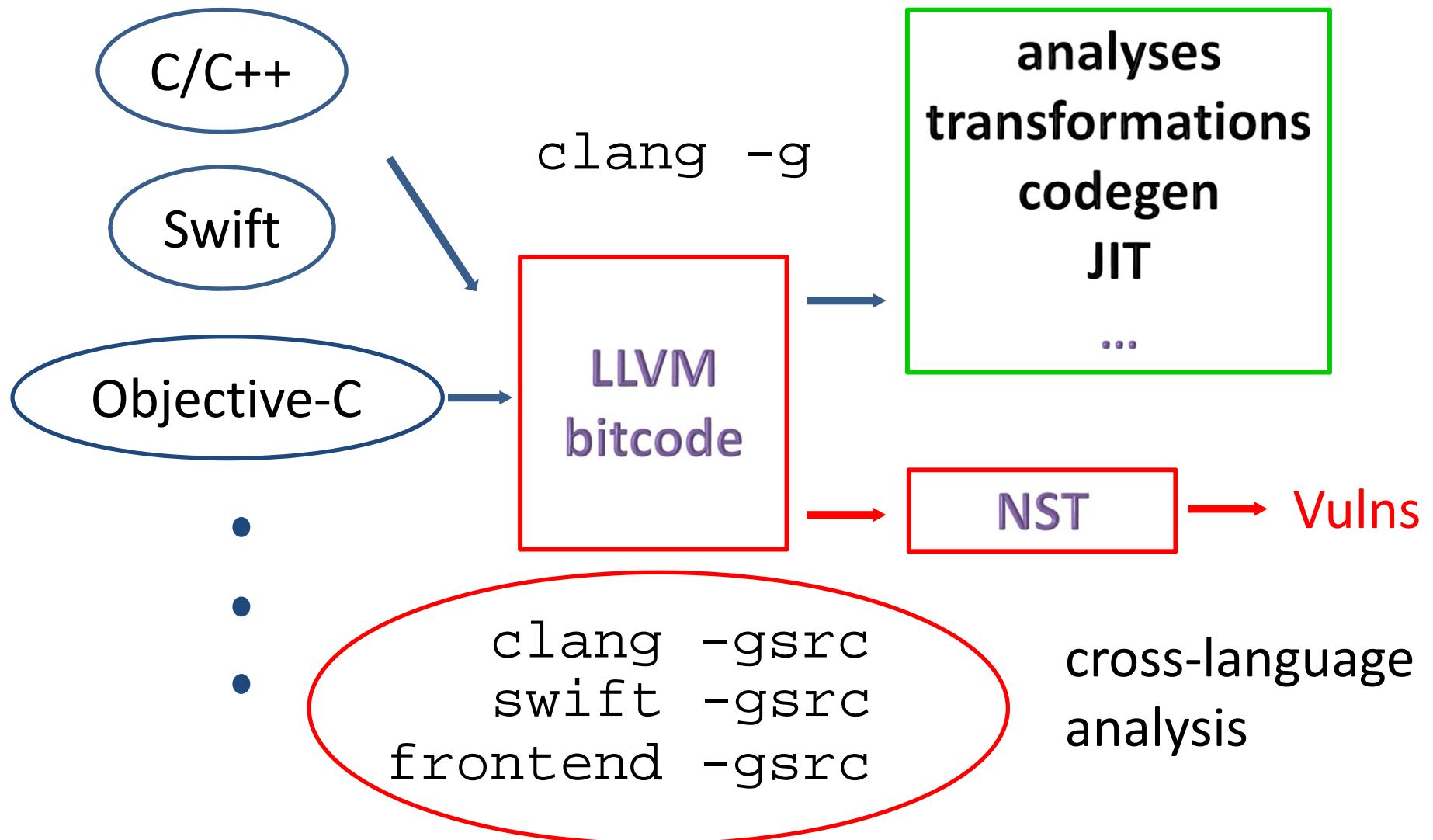
HP Fortify SCA for Objective-C



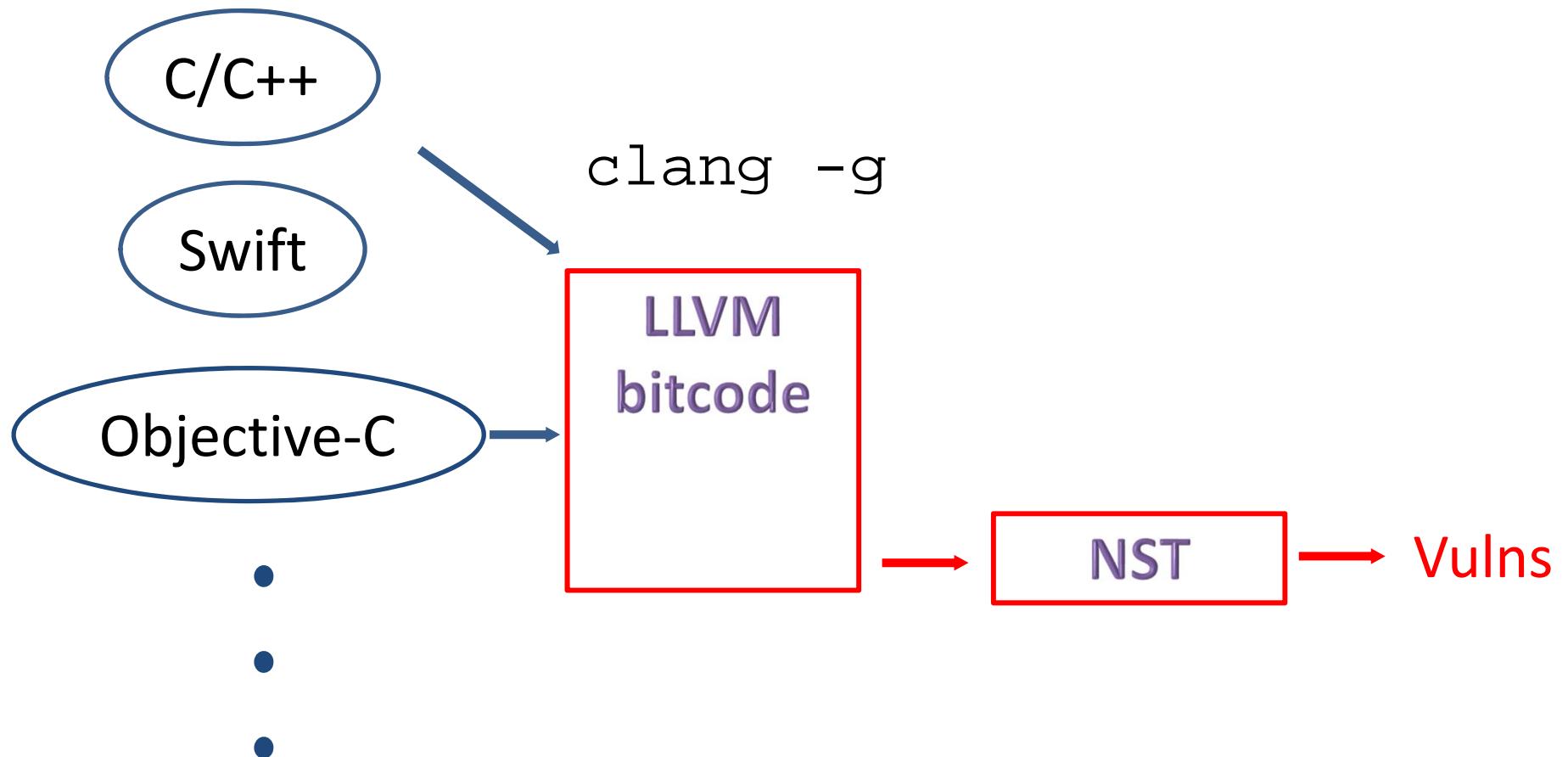
Bitcode with Enhanced Source Info



Bitcode with Enhanced Source Info



Why we cannot do this today?



Objective-C Static Taint Analyzer

```
@implementation HtmlViewController
- (void)viewDidLoad {
    if (_content) {
        ...
    } else {
        // Display the "About iGoat" splash screen as a default.
        ...
        NSString *fileContents =
[[NSString alloc] initWithContentsOfFile:filePath
encoding:NSUTF8StringEncoding error:&error];
        NSString *version = [[[NSBundle mainBundle] infoDictionary]
objectForKey:@"CFBundleShortVersionString"];
        [self.webView loadHTMLString:[NSString
stringWithFormat:fileContents, version] baseURL:baseURL];
    }
}
...
@end
```

Objective-C Static Taint Analyzer

```
@implementation HtmlViewController
- (void)viewDidLoad {
    if (_content) {
        ...
    } else {
        // Display the "About iGoat" splash screen as a default.
        ...
        NSString *fileContents = taint source by API doc
[[NSString alloc] initWithContentsOfFile:filePath
encoding:NSUTF8StringEncoding error:&error];
        NSString *version = [[[NSBundle mainBundle] infoDictionary]
objectForKey:@"CFBundleShortVersionString"];
        [self.webView loadHTMLString:[NSString
stringWithFormat:fileContents, version] baseURL:baseURL];
    }
    ...
}
@end
```

Objective-C Static Taint Analyzer

```
@implementation HtmlViewController
- (void)viewDidLoad {
    if (_content) {
        ...
    } else {
        // Display the "About iGoat" splash screen as a default.
        ...
        NSString *fileContents =
        [[NSString alloc] initWithContentsOfFile:filePath
encoding:NSUTF8StringEncoding error:&error];
        NSString *version = [[[NSBundle mainBundle] infoDictionary]
objectForKey:@"CFBundleShortVersionString"];
        [self.webView loadHTMLString:[NSString
stringWithFormat:fileContents, version] baseURL:baseURL];
    }
}
...
@end
```

taint sink by API doc

Objective-C Static Taint Analyzer

```

@implementation HtmlViewController
- (void)viewDidLoad {
    if (_content) {
        ...
    } else {
        // Display the "About iGoat" splash screen as a default.
        ...
        NSString *fileContents =
[[NSString alloc] initWithContentsOfFile:filePath
encoding:NSUTF8StringEncoding error:&error];
        NSString *version = [[[NSBundle mainBundle] infoDictionary]
objectForKey:@"CFBundleShortVersionString"];
        [self.webView loadHTMLString:[NSString
stringWithFormat:@"%@", fileContents, version] baseURL:baseURL];
    }
    ...
}
@end

```

taint source

taint sink

Objective-C Static Taint Analyzer

- Our taint source or taint sink is written in a declarative fashion, which is matched by the analyzer against its method signature.

NodeType: `TaintSource`

ClassName: `NSArray` | `NSString` | `NSData` |
`NSConstantString`

MethodSig: `arrayWithContentsOfFile:` |
`(string|init)WithContentsOfFile:(usedE|e)ncoding:error:` |
`| initWithContentsOfFile:` |
`(data|init)WithContentsOfFile:(options:error:)?`

Output: `return`

TaintFlags: `FILE_SYSTEM,XSS`

A Source-friendly IR

- A method signature

```
public class NSString extends NSObject {  
    public virtual NSString*  
initWithContentsOfFile$encoding$error$(  
                                NSString* this, ...);  
}
```

From Bitcode to Source

```
int convert(unsigned u) { return 0; }
```

```
define i32 @convert(i32 %u) #0 {
entry:
    ret i32 0
}
```

```
!4 = metadata !{i32 786478, metadata
!1, metadata !5, metadata !"convert",
metadata !"convert",...} ; [
DW_TAG_subprogram ] [line 25] [def]
[convert]
```

From Bitcode to Source

```
NamedMDNode *M_Nodes =
M->getNamedMetadata("llvm.debug.cu");
DIArray SPs = CU.getSubprograms();
for (unsigned i2 = 1,
         e2 = SPs.getNumElements();
         i2 != e2; ++i2) {
    DISubprogram DISP(SPs.getElement(i2));
    DICompositeType DIC(DISP.getType());
    DIArray Tys = DIC.getTypeArray();
    // Tys[0] return type
    // others are parameter types
}
```

No Metadata for Declarations

```
extern int convert(unsigned u);
```

```
declare i32 @convert(i32 %u) #2;
```

No metadata describing @convert.

No Metadata for Declarations

```
extern int convert(unsigned u);
```

```
declare i32 @convert(i32 %u) #2;
```

Metadata emission is a subprocess during code emission. **No code generation, no metadata.**

Generate Bitcode with Rich Source Info

- Decouple metadata emission and code generation.
- Control rich metadata emission by using -gsrc

```
$ clang -gsrc -O0 -c -emit-llvm -S  
HtmlViewController.m
```

Bitcode with Rich Source Info

```
declare extern_weak i8* @"-[NSString
initWithContentsOfFile:encoding:error:]"
(%1*, i8*, %1*, i64, %3**)

!1538 = metadata !{i32 786478, metadata
!4, metadata !302, metadata !"-[NSString
initWithContentsOfFile:encoding:error:]"
, . . . } ; [ DW_TAG_subprogram ] . . .
```

Bitcode with Rich Source Info

```
Type signature: (NSString*,  
objc_selector*, NSString*,  
NSStringEncoding, NSError** ) ->  
NSString*
```

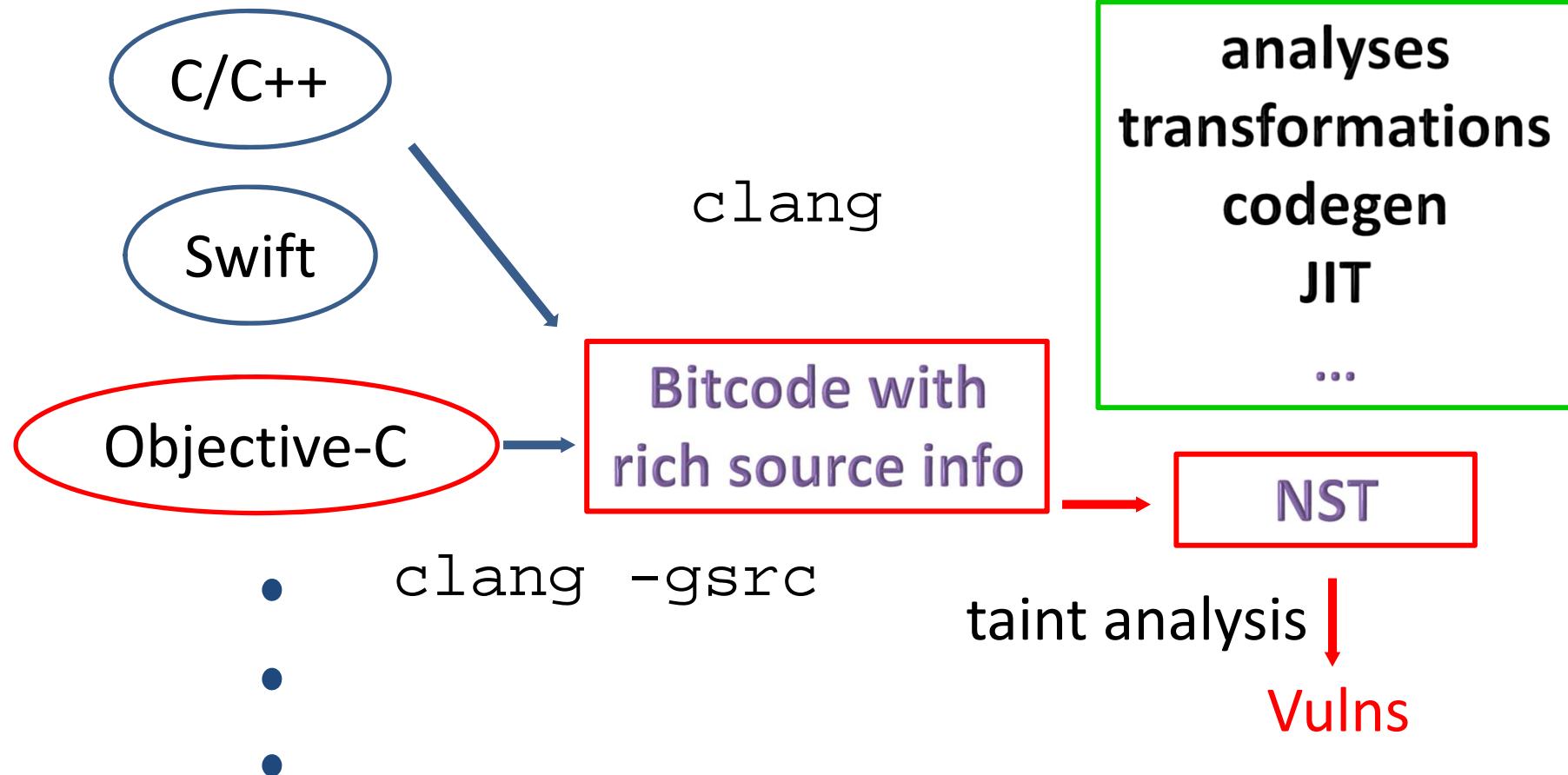
```
typedef: NSStringEncoding,  
NSUInteger,  
long unsigned int
```

A Source-friendly IR

- NST

```
public class NSString extends NSObject {  
    public virtual NSString*  
initWithContentsOfFile$encoding$error$(  
                                NSString* this, ...);  
}
```

Bitcode with Enhanced Source Info



Small Modification Big Opportunity

- Entire patch to Clang/LLVM has 543 lines for 3.3 (git diff)
- Upgrading to 3.5

Small Modification Big Opportunity

- All frontends should implement this feature

