MWR Labs

++

Bug Hunting with Static Code Analysis

Nick Jones

6th June 2016

MWR LABS

The Problem

+ Software developers make mistakes
+ Mistakes = bugs = vulnerabilities
+ Our goal is fewer bugs

++

Who am I?

Nick Jones

+ Security Consultant at MWR InfoSecurity
+ Web application security, infrastructure assessments
+ Previous experience doing commercial software development
+ Developed bespoke analysis tools for clients

++

What will we be covering?

+ The problem of applications security
+ Regular Expressions
+ Parsers
+ Control Flow Graphs
+ Case study: bug hunter
+ Case study: software developer

++

What will we be covering?

+ The problem of applications security
+ Regular Expressions
+ Parsers
+ Control Flow Graphs
+ Case study: bug hunter
+ Case study: software developer

MWR LABS

A Case Study

+ MWREvents has developed a new online events planning platform – website and mobile apps
+ Their developers are of average quality
+ No in-house security experts
+ Want to find and fix all their security issues

++

How Do We Find Bugs?

Static Analysis

+ Analysing an application without executing it
+ Code review, binary analysis, reverse engineering

Dynamic Analysis

+ Analysing by monitoring and interacting with the application as it executes
+ Fuzzing, tampering, functional testing

++

How Do We Find Bugs?

Static Analysis

+ Analysing an application without executing it
+ Code review, binary analysis, reverse engineering

Dynamic Analysis

+ Analysing by monitoring and interacting with the application as it executes
+ Fuzzing, tampering, functional testing

++

How Do We Find Bugs?

Static Analysis

+ Analysing an application without executing it
+ Code review, binary analysis, reverse engineering

Dynamic Analysis

+ Analysing by monitoring and interacting with the application as it executes
+ Fuzzing, tampering, functional testing

++

How Do We Code Review?

Manual

+ Give code to smart security experts
+ They read, understand and spot bugs

Automated

+ Pass code to tool
+ Tool parses code, hunts for known issues

MWR LABS

Code Review – Examples

```
void echo ()
{
    char buf[8];
    gets(buf);
    printf("%s\n", buf);
}
```

MWR LABS

++

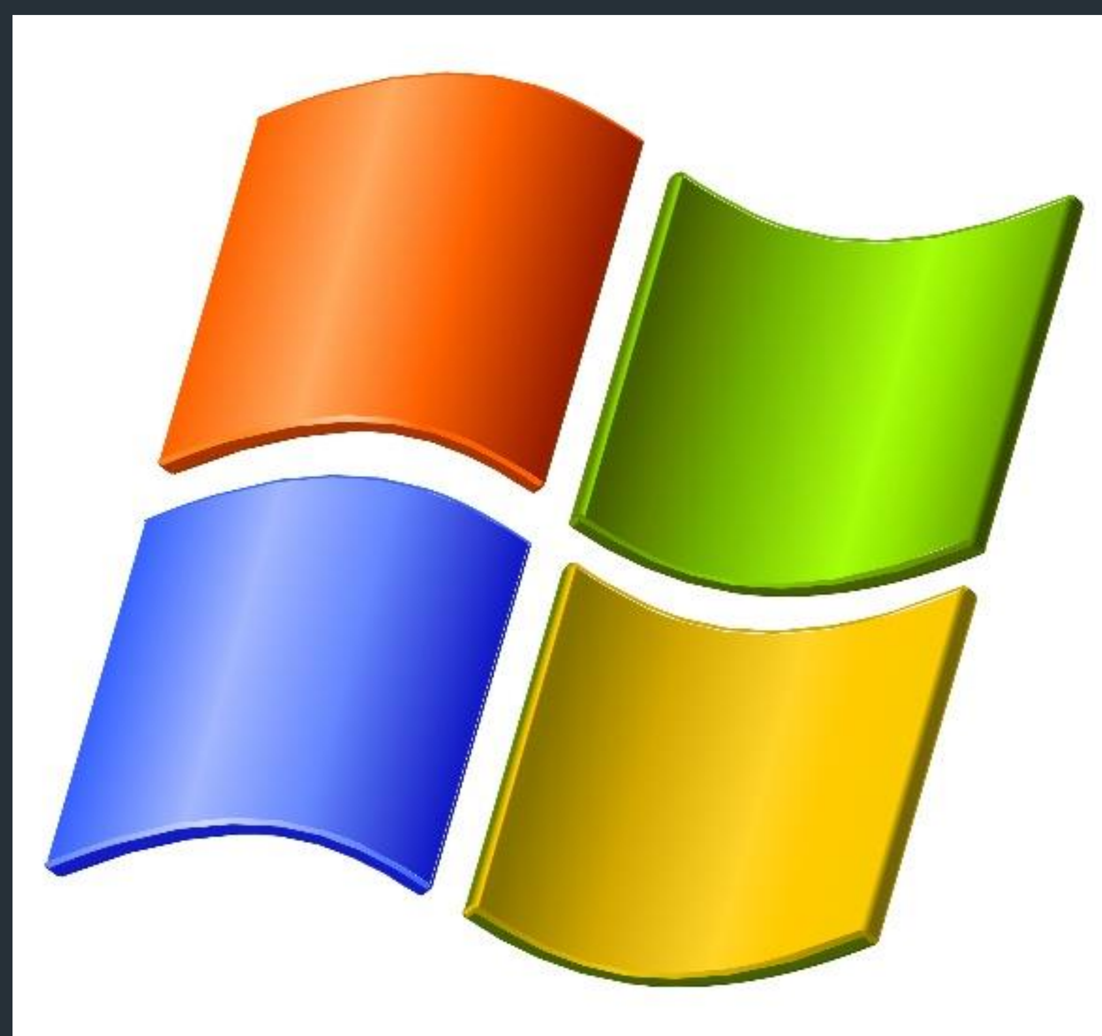
Code Review – Examples

```
webView.getSettings().setJavaScriptEnabled(true);
```

++

Manual Code Review – The Downsides

+ Manual code review is expensive



~45 Million LOC

~86 Million LOC

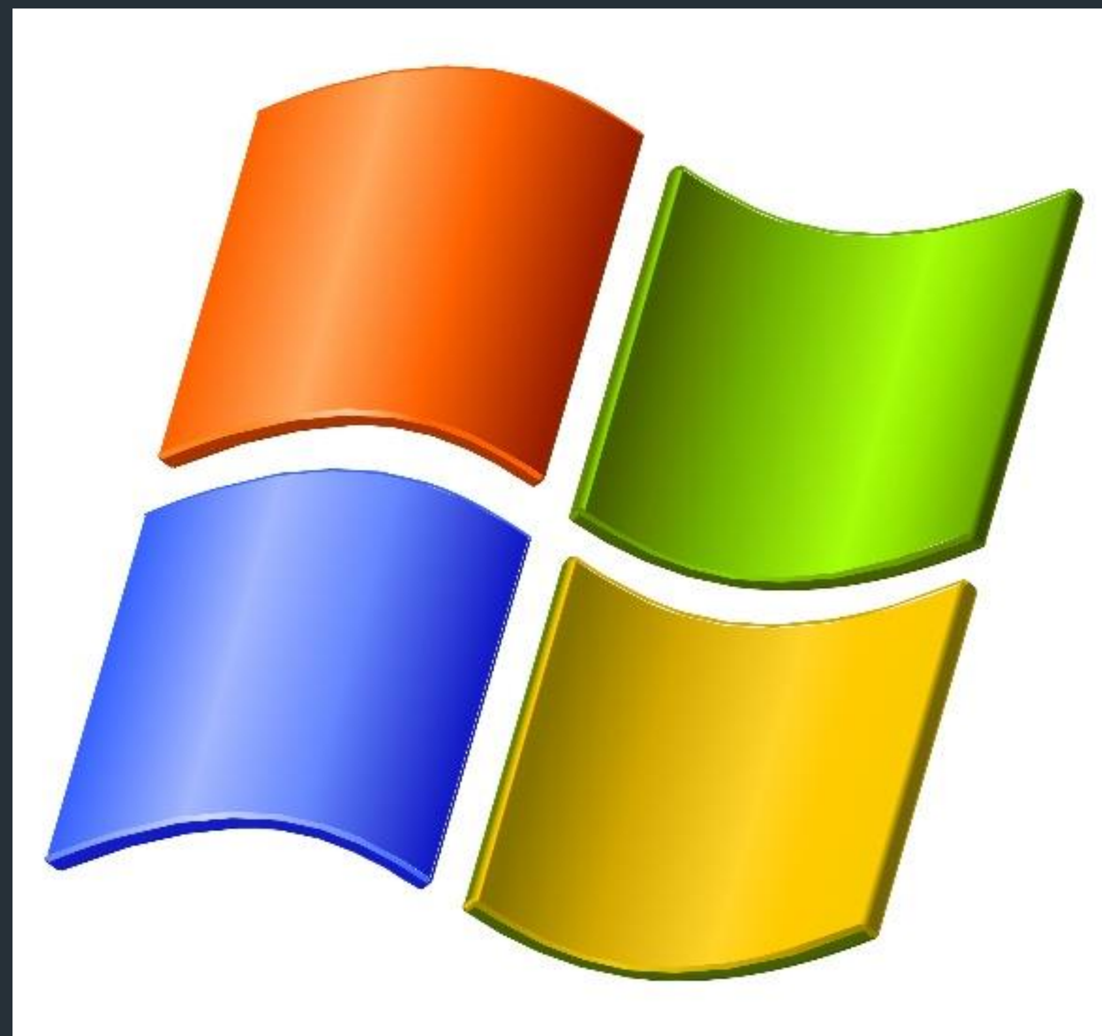
~24 Million LOC

MWR LABS

++

How Many Bugs Is That?

+ Steve McConnell (Code Complete) says 10-20 defects per 1000 lines of code



~675,000 bugs

~1,290,000 bugs

~360,000 bugs

++

Static Code Analysis

Automated searching of source code for issues

- + Higher up front costs
- + ‘Free’ security once built and configured
- + Catch low hanging fruit automatically

++

Computer Science Theory Ahead

To best use tools, you need to understand them.

+ Language types
+ Automata
+ Parsers

++

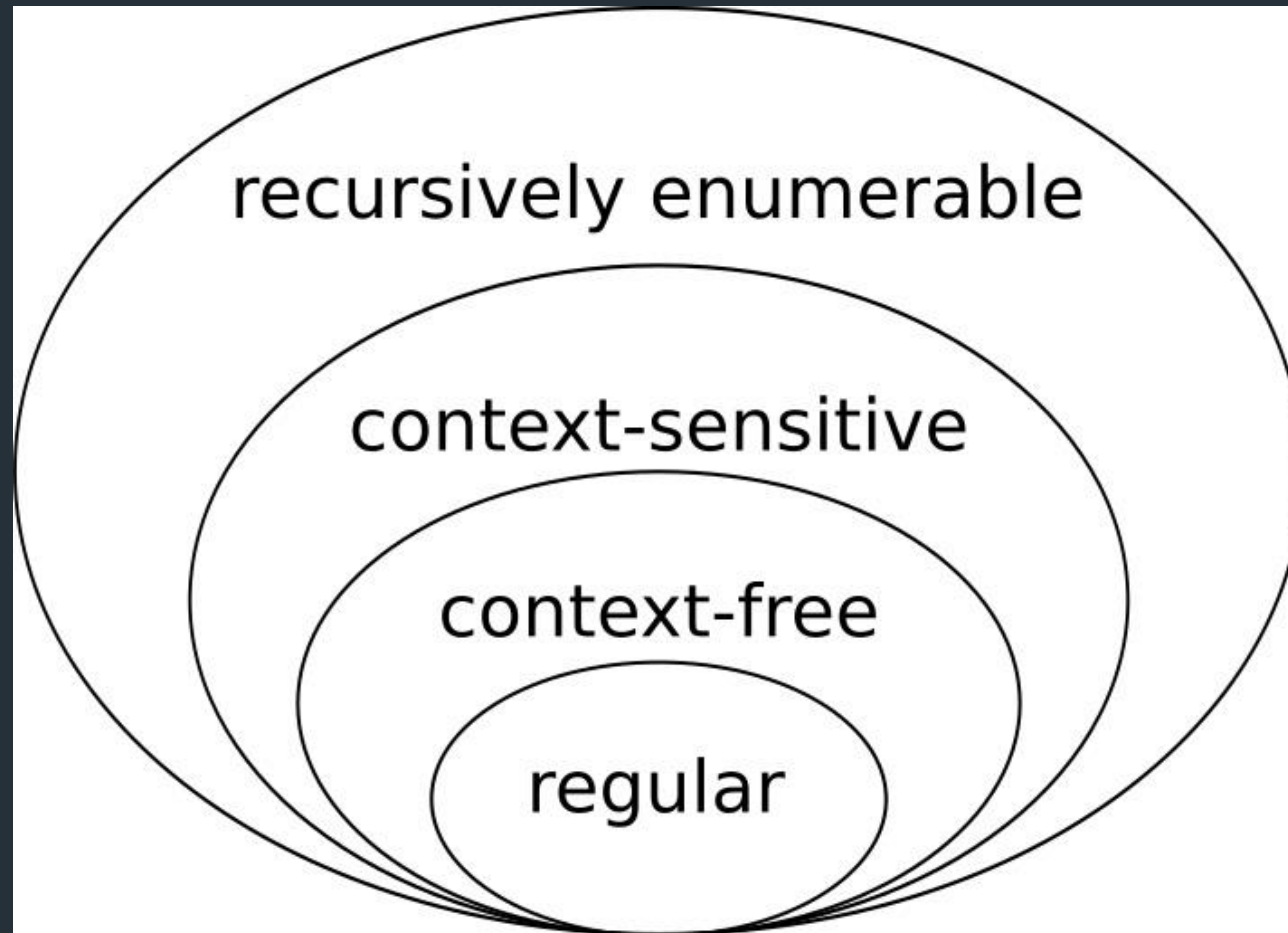
Languages

- "[A] set of strings of symbols that may be constrained by rules that are specific to it"
- Defined by a grammar

MWR LABS

++

Chomsky's Language Hierarchy



++

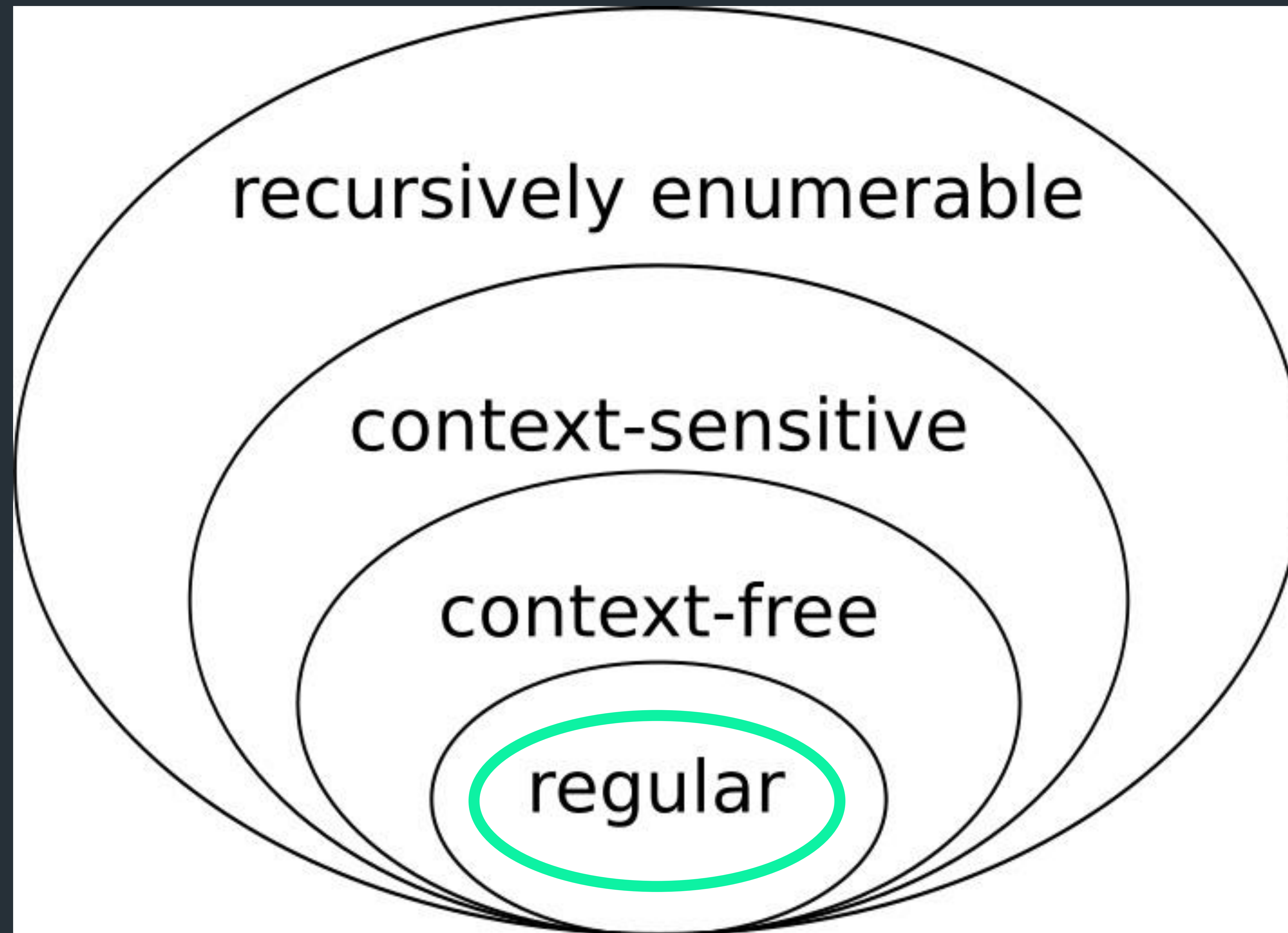
What will we be covering?

+ The problem of applications security
+ Regular Expressions
+ Parsers
+ Control Flow Graphs
+ Case study: bug hunter
+ Case study: software developer

MWR LABS

++

Chomsky's Language Hierarchy



++

Regular Expressions

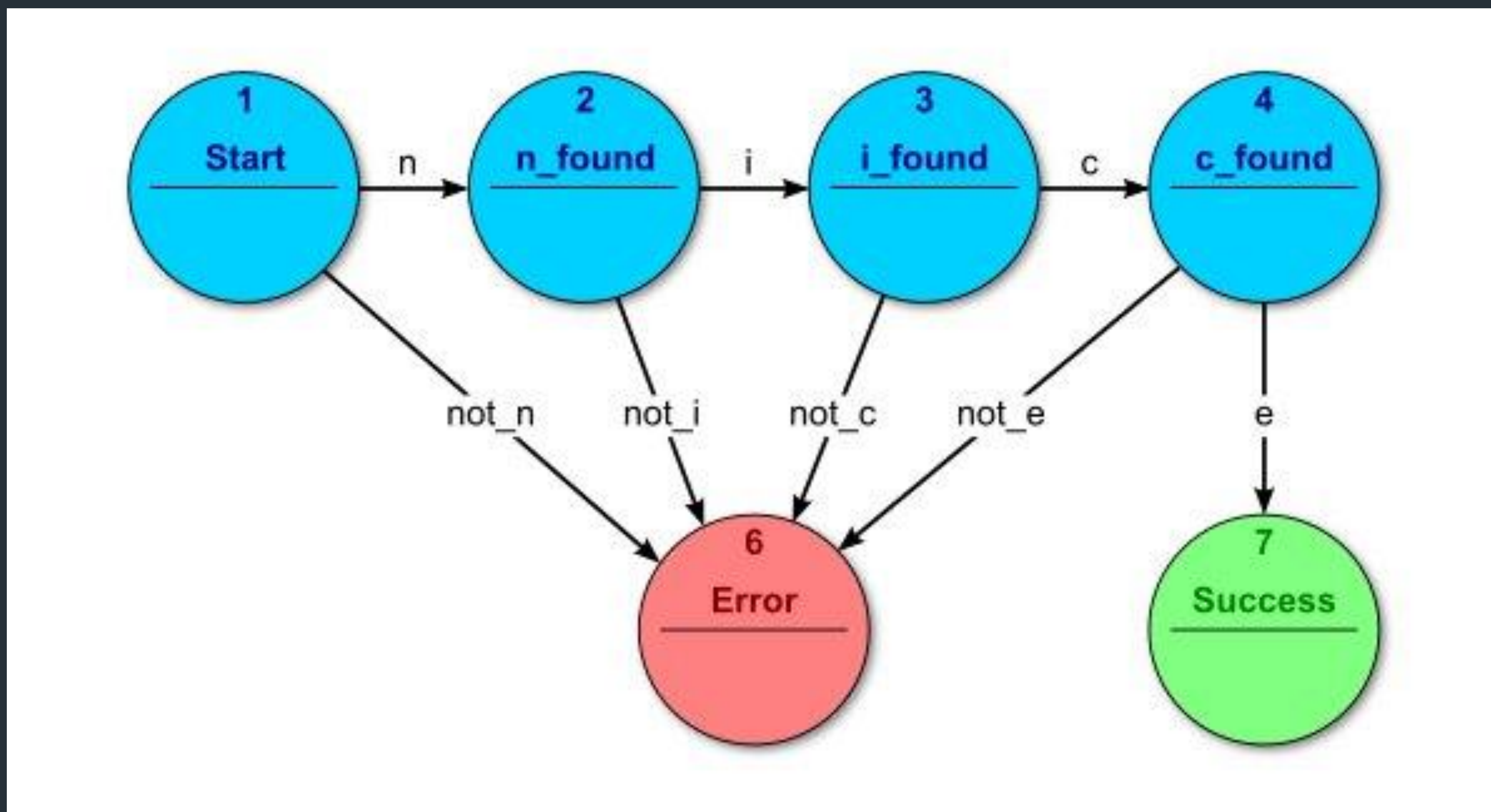
Regular expressions can parse any regular language

+ Act as a finite automata
+ List of states, list of transitions between them
+ Process input until accept or error state is reached

In practice, modern regexes are far more powerful than the definition given here, but the key limitations remain

MWR LABS

++
Regular Expressions



++

Bug Hunting with Regular Expressions

Match code snippets that look like known problems

+ Quick and easy to write, so low cost
+ "Does my code match this very specific known issue?"

+ Bad imports
+ Calls to known dangerous functions
+ Known security misconfigurations

++

Code Review – Examples

Code:

```
webView.getSettings().setJavaScriptEnabled(true);
```

Regex:

```
‘setJavaScriptEnabled\(true\)’
```

++

Code Review – Examples

Code:

```
webView.getSettings().setJavaScriptEnabled(true);
```

Regex:

```
'setJavaScriptEnabled\(true\)'
```

MWR LABS

++

Regular Expressions – Example

Code:

```
if (DEBUG) {
   printf('Debug statement 1: %s', var1);
   printf('Other stuff: %s', var1);
   printf('Finally: %s', var1);
}
```

Regex:

‘printf\(.*\)’

++

Regular Expressions – Example

Code:

```
if (DEBUG) {
   printf('Debug statement 1: %s', var1);
   printf('Other stuff: %s', var1);
   printf('Finally: %s', var1);
}
```

Regex:

‘printf\(.*\)’

++

Regular Expressions – Example

Code:

```
if (DEBUG) {
   printf('Debug statement 1: %s', var1);
   printf('Other stuff: %s', var1);
   printf('Finally: %s', var1);
}
```

Regex:

‘printf\(.*\)’

MWR LABS

++

Regular Expressions – Example

Code:

```
if (DEBUG) {
   printf('Debug statement 1: %s', var1);
   printf('Other stuff: %s', var1);
   printf('Finally: %s', var1);
}
```

Regex:

‘printf\(.*\)’

++

Regular Expressions – The Disadvantages

Regular expressions can’t ‘count’

- No way to maintain state
- Cannot back trace

++

Regular Expressions – The Disadvantages

Two options to check for debug guard:

+ Check backwards line by line until you reach beginning of file – inefficient
+ Check X many previous lines – lots of false positives

Three alerts generated for the same missing guard

++

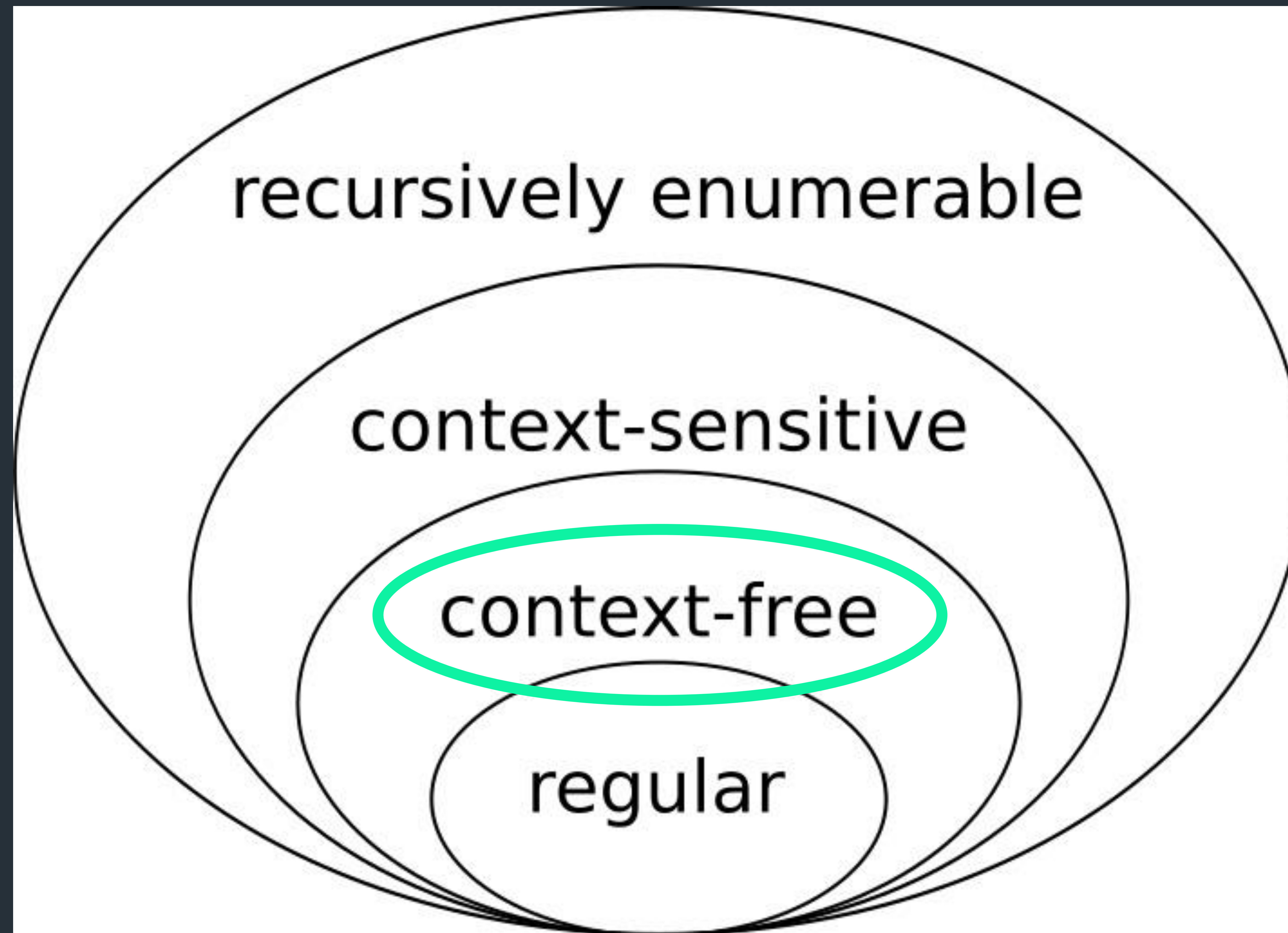
Regular vs Context-Free Languages

+ Regular expressions only match regular languages*
+ Programming languages usually context-free

*mostly

MWR LABS

++
Chomsky’s Language Hierarchy



++

What will we be covering?

- The problem of applications security
- Regular Expressions
- Parsers
- Control Flow Graphs
- Case study: bug hunter
- Case study: software developer

++

Context-Free Languages

+ Superset of regular languages
+ Anything that can be accepted by a pushdown automata

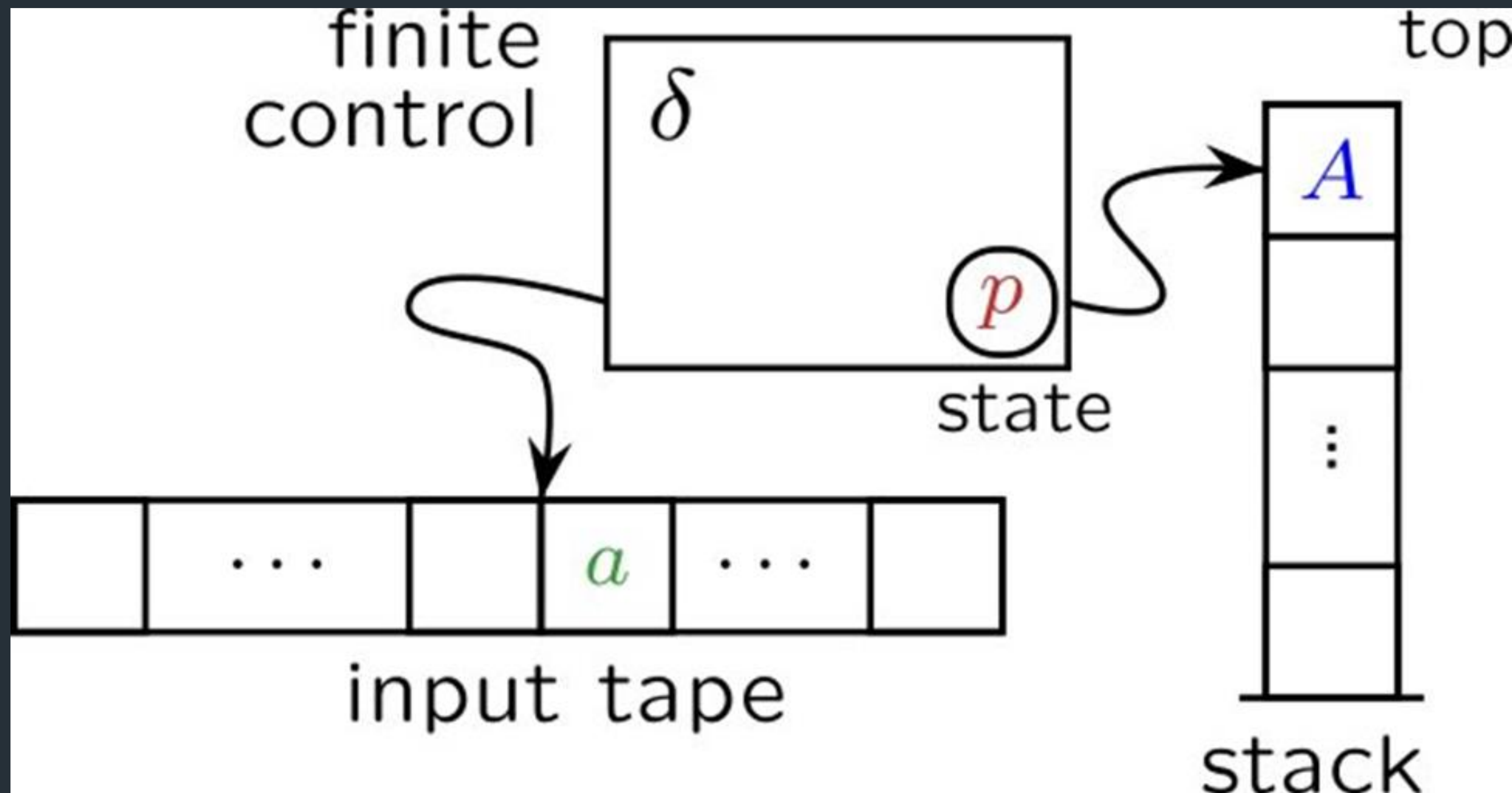
++
Pushdown Automata

+ Finite State Machines with stacks
+ Decide transition based on both input and top of stack
+ Can push/pop to stack as needed

MWR LABS

++

Pushdown Automata



++

Parsers

- Converts text into a hierarchical data structure
- Several different types, depending on what you're parsing
- TL;DR: Construct a Parse Tree or Abstract Syntax Tree (AST) from the source code

++

Parsers

Two separate stages

+ Lexer splits input text into tokens (strings with an understood meaning)
+ Parser constructs AST or similar from list of tokens

Can combine both – scannerless parsing

++

Lexer Example

Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```

Lexed Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```

++

Lexer Example

Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```

Lexed Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```

++

Parser Example

Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```

MWR LABS

++

Parser Example

Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```

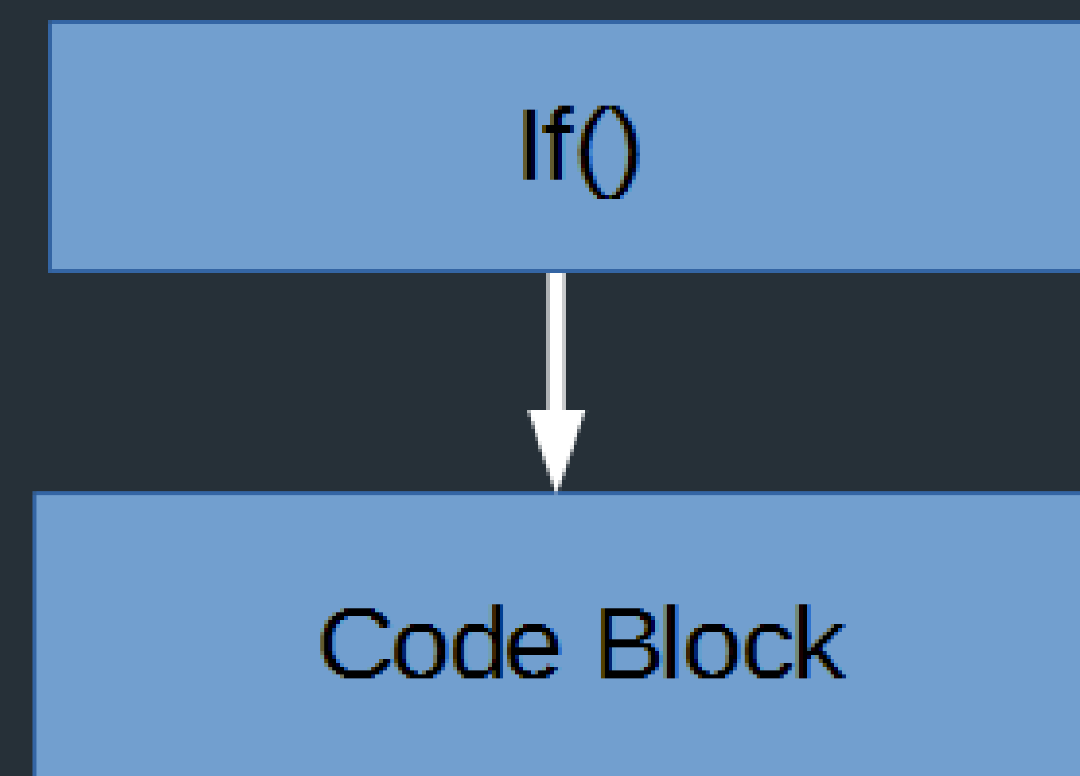
If()

MWR LABS

++
Parser Example

Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```



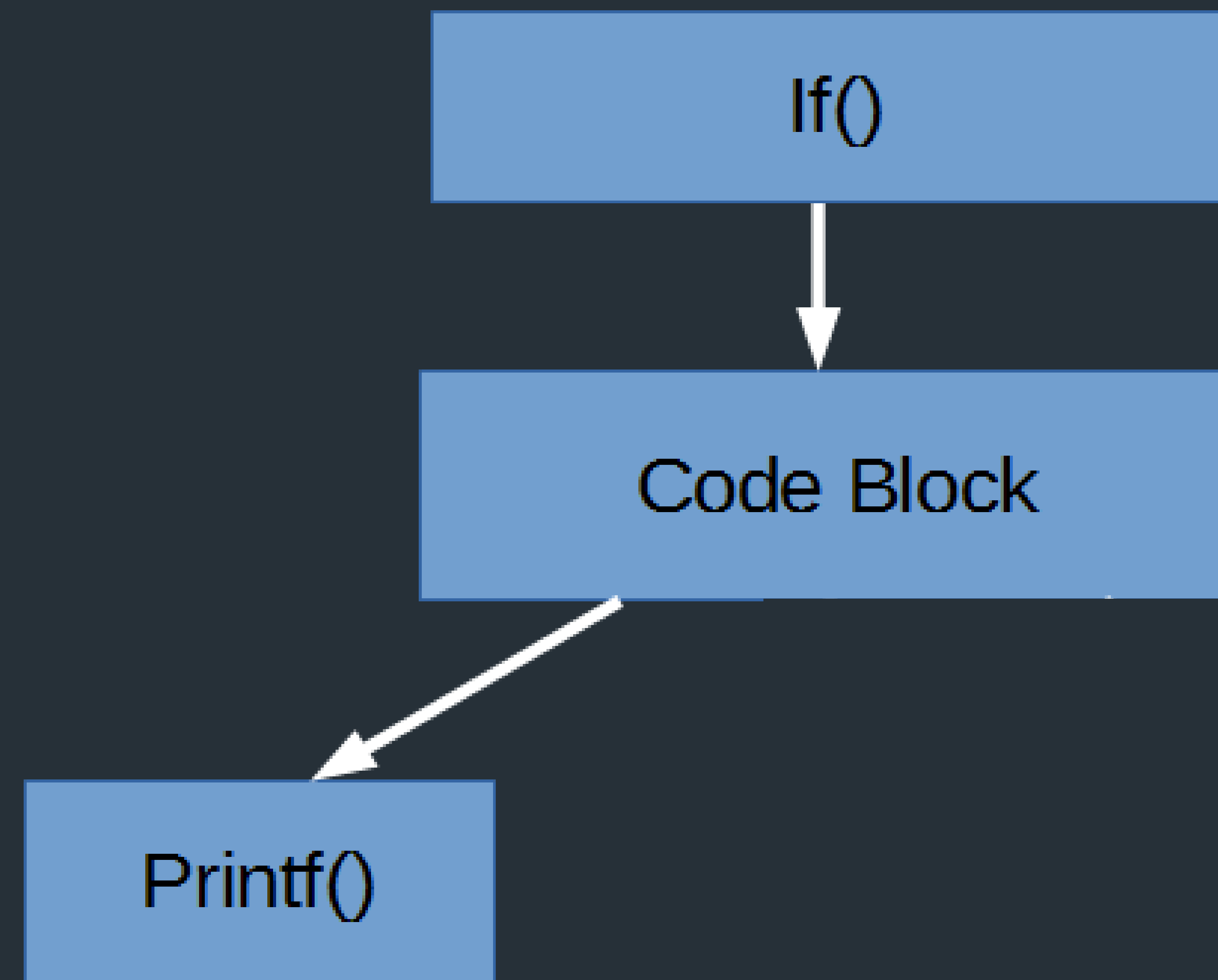
MWR LABS

++

Parser Example

Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```



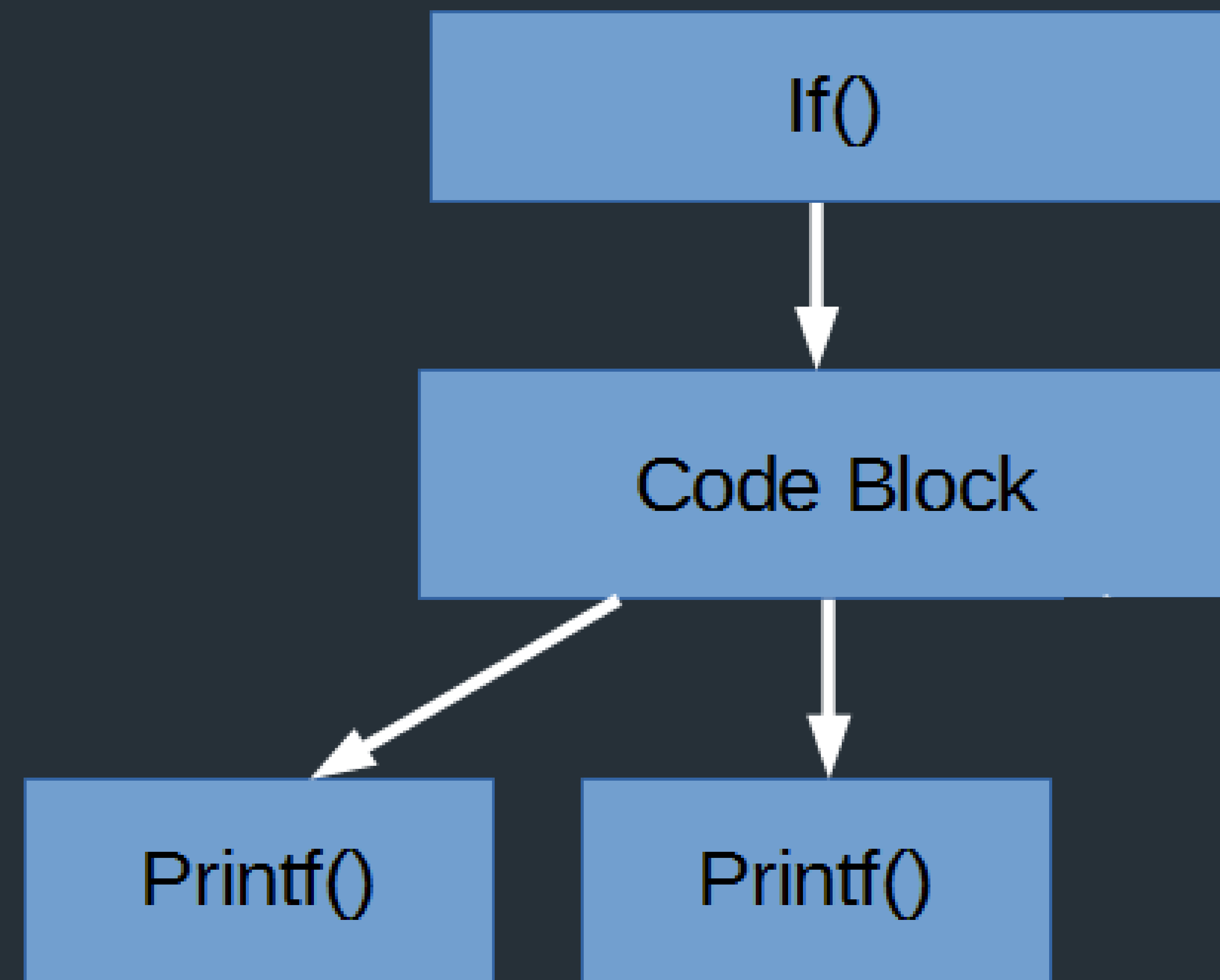
MWR LABS

++

Parser Example

Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```



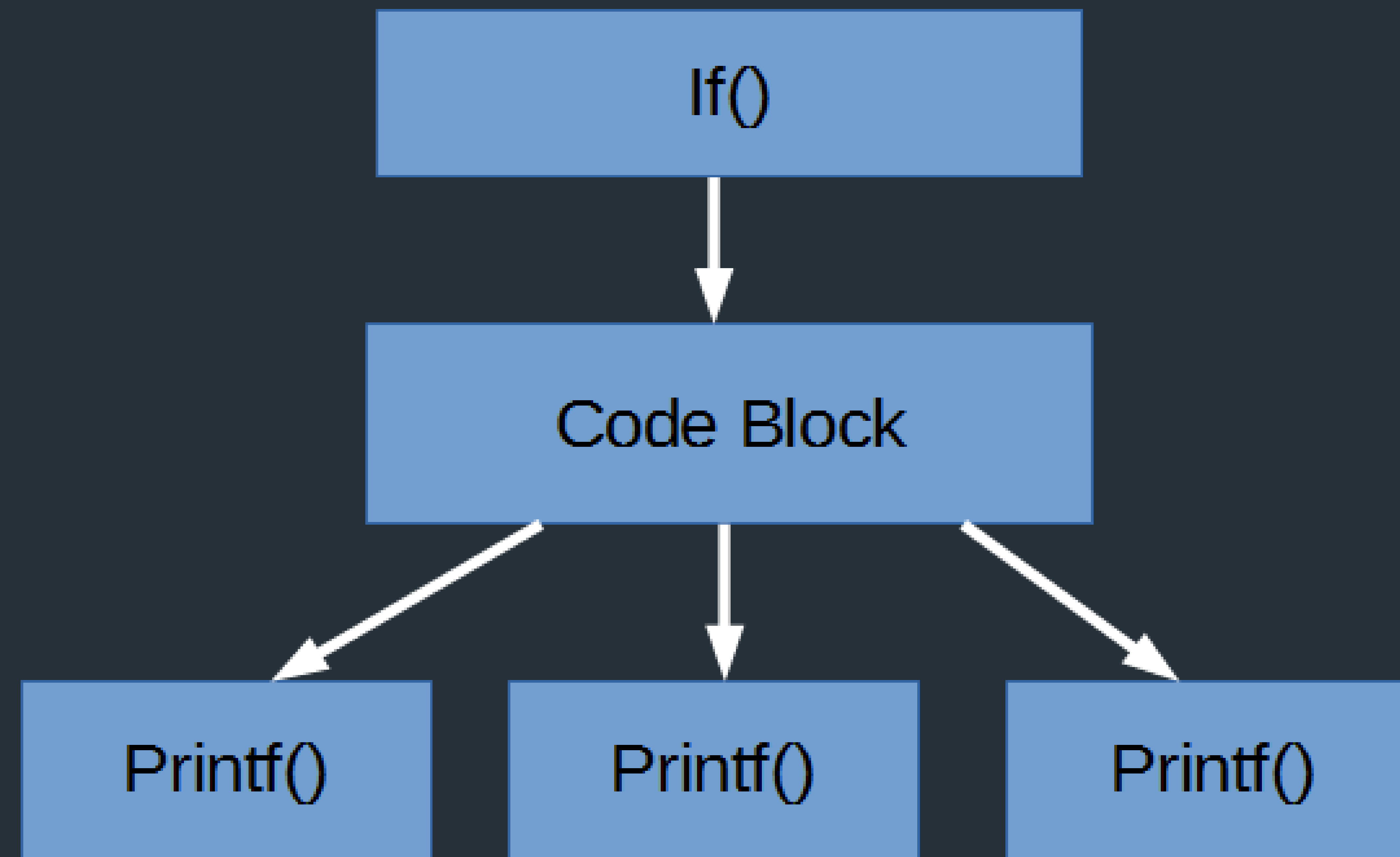
MWR LABS

++

Parser Example

Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```



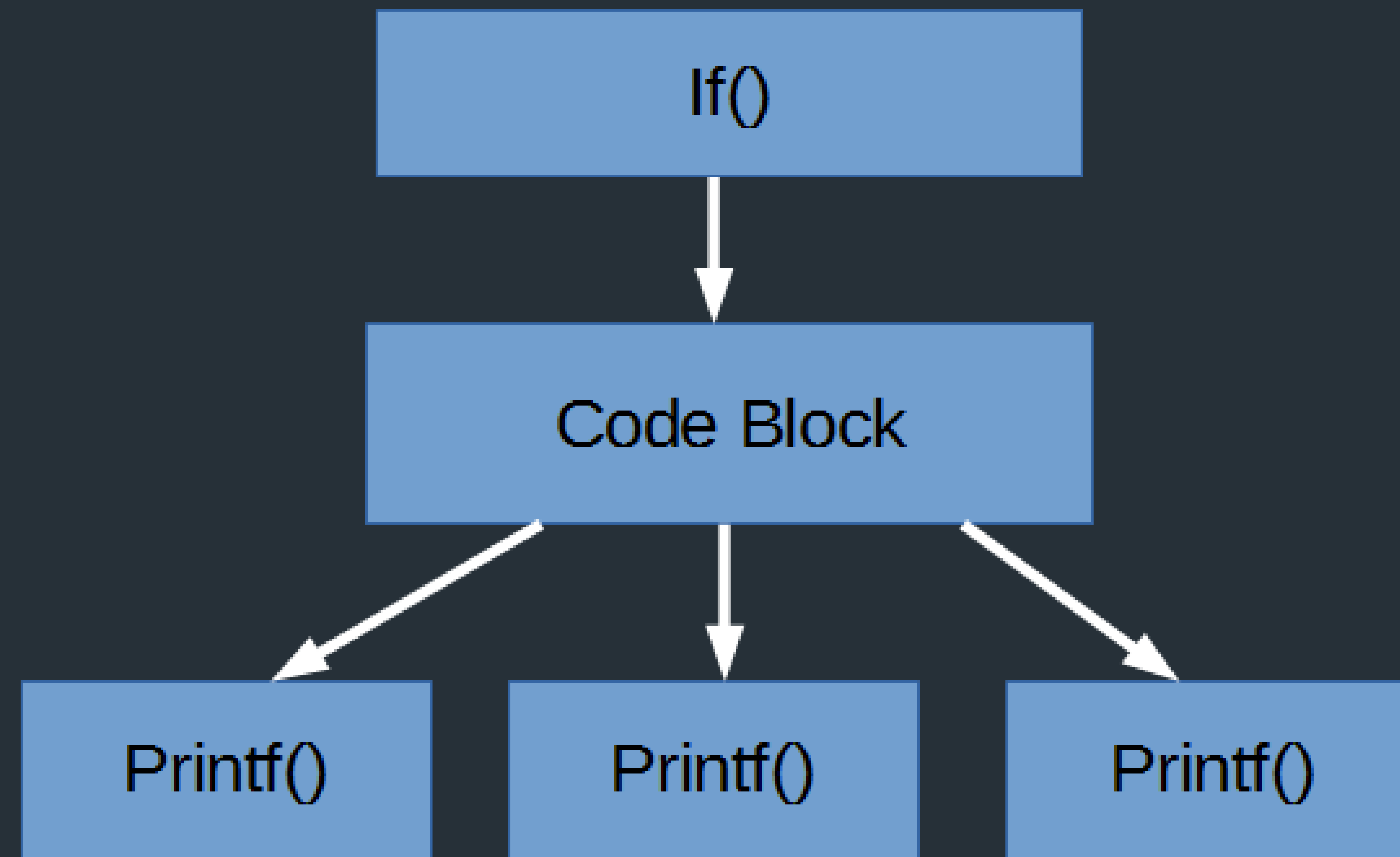
MWR LABS

++

Parser Example

Code:

```
if (DEBUG)
{
   printf(…);
   printf(…);
   printf(…);
}
```



MWR LABS

++

We've got an AST, now what?

Basic:

+ Search AST for dodgy function calls, check for guards
+ Check for questionable imports
+ Same as before, fewer false positives

Advanced:

+ Control Flow Graphs (CFGs)
+ Taint Analysis

MWR LABS

++

What will we be covering?

+ The problem of applications security
+ Regular Expressions
+ Parsers
+ Control Flow Graphs
+ Case study: bug hunter
+ Case study: software developer

MWR LABS

++

Control Flow Graphs

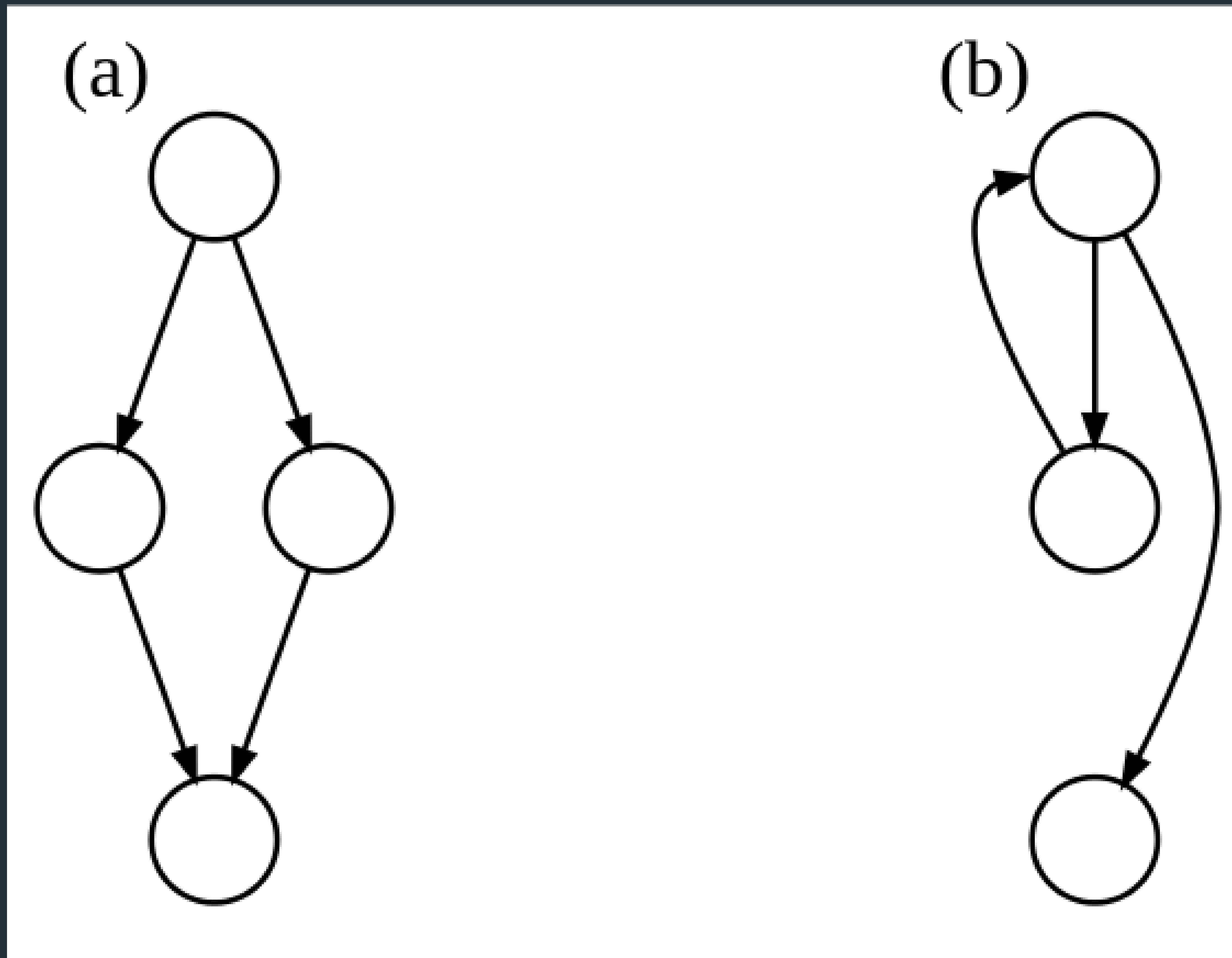
“a representation, using graph notation, of all paths that might be traversed through a program”

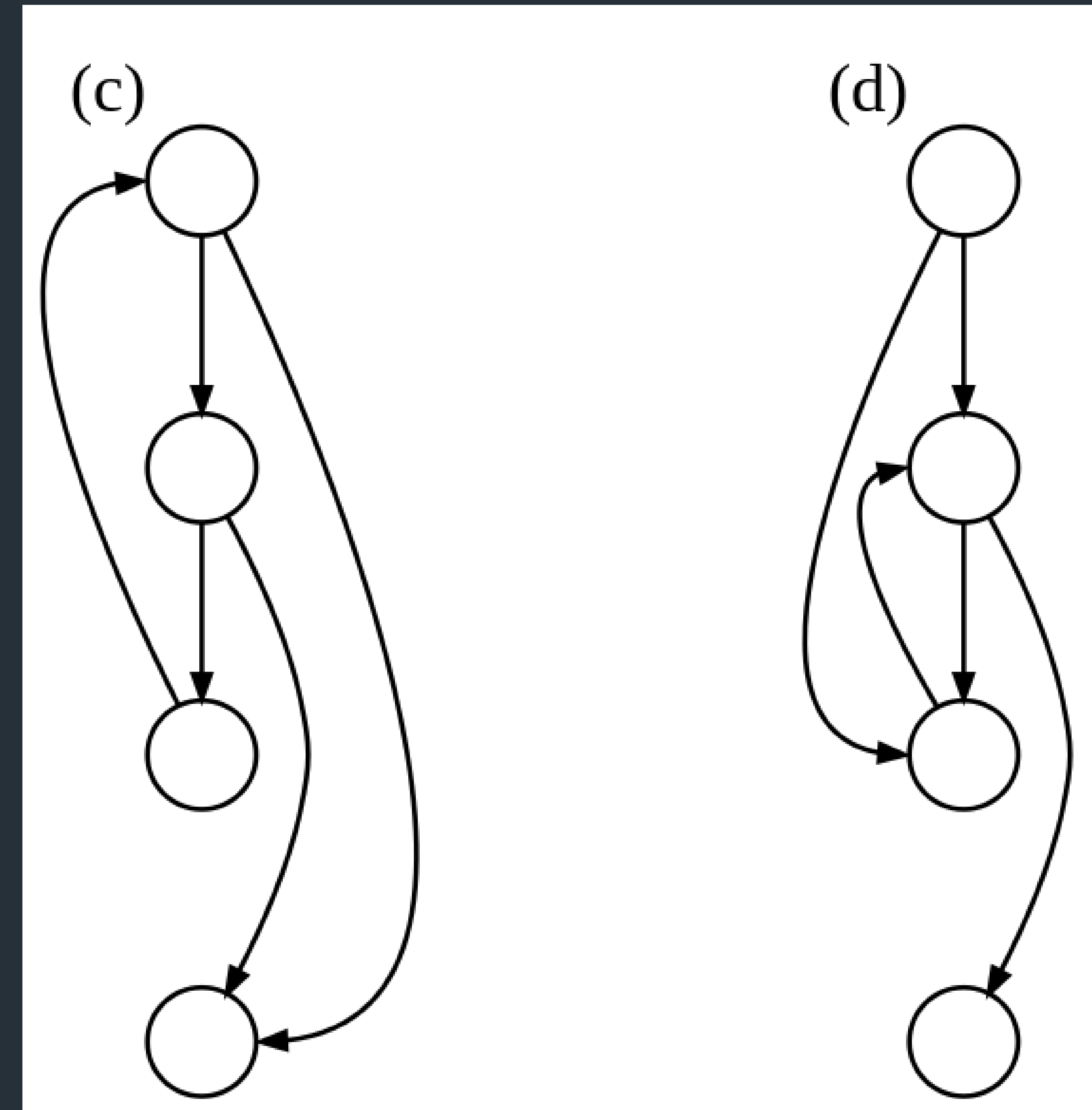
- Each basic block represented as a graph node
- Jump targets start block, jumps end block
- Jumps represented as directed edges

MWR LABS

++

Control Flow Graphs





++

Why Should I Care About Control Flow Graphs?

- Allows tracing of execution dependant on given inputs without running the application
- Trace data sinks back to original source
- Data sanitized several function calls ago? Trace the graph back and find it

++

Why Should I Care About Control Flow Graphs?

```
$result = login($_POST['user'], $_POST['password']);

function login(user, password) {
   return login_query(user, password);
}

function login_query(user, password) {
   return mysqli_query('select * from user where
user=' + $user + ' and password=' + $password + ';');
}
```

++

Why Should I Care About Control Flow Graphs?

```
$result = login($_POST['user'], $_POST['password']);

function login(user, password) {
    return login_query(user, password);
}

function login_query(user, password) {
    return mysqli_query('select * from user where
user=' + $user + ' and password=' + $password + ';');
}
```

++

Why Should I Care About Control Flow Graphs?

```
$result = login($_POST['user'], $_POST['password']);

function login(user, password) {
   return login_query(user, password);
}

function login_query(user, password) {
   return mysqli_query('select * from user where
user=' + $user + ' and password=' + $password + ';');
}
```

++

Why Should I Care About Control Flow Graphs?

```
$result = login($_POST['user'], $_POST['password']);

function login(user, password) {
   return login_query(user, password);
}

function login_query(user, password) {
   return mysqli_query('select * from user where
user=' + $user + ' and password=' + $password + ';');
}
```

MWR LABS

++

Why Should I Care About Control Flow Graphs?

```
$result = login($_POST['user'], $_POST['password']);

function login(user, password) {
   return login_query(user, password);
}

function login_query(user, password) {
   return mysqli_query('select * from user where
user=' + $user + ' and password=' + $password + ';');
}
```

++

Why Should I Care About Control Flow Graphs?

```
$result = login($_POST['user'], $_POST['password']);

function login(user, password) {
   return login_query(user, password);
}

function login_query(user, password) {
   return mysqli_query('select * from user where 
user=' + $user + ' and password=' + $password + ';');
}
```

MWR LABS

++

Parsers

Downsides:

+ Higher upfront cost to develop
+ More computationally intensive

++

The Bigger Picture

These tools all fit into a larger picture, all of which needs to work together

+ Static code analysis
+ Manual code review
+ Fuzzing
+ Functional testing

++

What will we be covering?

+ The problem of applications security
+ Regular Expressions
+ Parsers
+ Control Flow Graphs
+ Case study: bug hunter
+ Case study: software developer

++

Case Studies

Two primary categories of people:

+ Bug hunters – security consultants, people doing bug bounties or looking for 0-days
+ Developers – people building applications who care about security

++

I’m a bug hunter, why do I care?

+ Target identification – pick a project to go after
+ Find low hanging fruit
+ Identify ropey parts of the codebase

++

Target Identification

- Download source for a bunch of projects
- Run analyser on all of them, look at the outputs

++

Target Identification – Example

	OpenSSL	LibreSSL	GnuTLS	mbedTLS
Flawfinder	1794	1389	1228	1381

MWR LABS

++

Target Identification – Example

./src/pkcs11.c:871: [4] (buffer) strcpy: Does not check for buffer overflows when copying to destination. Consider using strncpy or strlcpy (warning, strncpy is easily misused).

++

Low Hanging Fruit

+ SQL Injection
+ XSS
+ Buffer Overflows
+ Some Use after Frees

++

Low Hanging Fruit

SQL Injection, XSS, Buffer Overflows

+ Look for data sinks – SQL queries, user-provided data rendering etc
+ Trace input to data sinks back up CFG to source
+ If no sanitisation on user-provided data, probably an attack vector

MWR LABS

++

Low Hanging Fruit

Use after frees

- Track allocation/deallocation of pointers through CFG
- UAF where pointer referenced after deallocation

++

Example Tools

+ Flawfinder (C/C++)
+ Graudit (ASP/C/.NET/JSP/Perl/PHP/Python)
+ Find Security Bugs (Java, FindBugs Plugin)
+ RATS (C/C++/Perl/PHP/Python)
+ RIPS (PHP)
+ Brakeman (Ruby/Rails)

++

Example Libraries/Platforms

For building your own:

+ Clang Analyzer
+ PLY and libraries that build on it (PLYJ for Java)
+ Pyparsing
+ ANTLR
+ Coco/R

++

What will we be covering?

- The problem of applications security
- Regular Expressions
- Parsers
- Control Flow Graphs
- Case study: bug hunter
- Case study: software developer

++

Static Analysis for Developers

+ Catch security issues before the penetration tests
+ One developer builds it, everyone can use it
+ Can be built into existing toolchains and development lifecycles

MWR LABS

++
Static Analysis and CI

+ CI: Continuous Integration
+ Continuously integrating new features as they're developed
+ Periodic automated compilation and testing

++

CI Tooling Examples

+ Hudson
+ Jenkins
+ Travis CI
+ Bamboo
+ Team Foundation Server

++

CI Workflow

- Developer checks in code
- Server compiles code
- Test suites are automatically run

++
CI Workflow

Jenkins

search

Jenkins › robot ›

ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure
- Robot Results
- Email Template Testing

Build History (trend)

#6	Aug 20, 2013 2:26:16 PM
#5	Aug 20, 2013 2:24:49 PM
#3	Aug 20, 2013 2:14:06 PM
#2	Aug 15, 2013 4:24:27 PM
#1	Aug 15, 2013 4:21:47 PM

RSS for all RSS for failures

Project robot

add description

Disable Project

Workspace

Recent Changes

Latest Test Result (no failures)

Latest Robot Results:

	Total	Failed	Passed	Pass %
Critical tests	10	0	10	100.0
All tests	10	0	10	100.0

> Browse results
> Open smoke_all_report.html
> Open smoke_all_log.html

Robot Framework Tests Trend (all tests)

> Zoom graph to changes

Number of testcases: 0, 2, 4, 6, 8, 10

Build number: #1, #2, #3, #5, #6

Passed
Failed

Permalinks

- Last build (#6), 20 hr ago
- Last stable build (#6), 20 hr ago
- Last successful build (#6), 20 hr ago

++

CI Advantages

+ Automated security testing
+ Catch issues as they are introduced to the codebase
+ Catch regressions in code before it hits production
+ Runs automatically, no developer interaction required

++

CI – Benefits

Case study – M&S data breach, Oct 2015

+ Developer error led to users being presented with other people's data on login
+ Personal details and partial card numbers exposed
+ Automated regression testing as part of CI would likely catch this

++

Commercial Static Analysis Tools

+ Veracode
+ Coverity
+ Fortify
+ Checkmarx
+ Klocwork

++

Commercial Tools

Coverity® Connect | dave_cid161026 | Configuration | Help | Simon Marriott | Enter CID(s)

DASHBOARDS
- Quality Advisor
- Security Advisor
- Test Advisor

ISSUES
- All Newly Detected
- All Untriaged
- High Impact Outstanding
- Low Impact Outstanding
- Medium Impact Outstanding
- My Newly Detected
- My Outstanding
- Newly Fixed Defects
- Outstanding Defects
- Outstanding Security Risks
- Outstanding Test Rules Violatio...
- Unassigned
- coverity_gerrit
- simonmarriott
- spl_report
- Recently Viewed

FILES
- Covered By Tests
- In Latest Snapshot
- Uncovered By Tests
- With Outstanding Issues
- With Untriaged Issues

FUNCTIONS
- Covered By Tests
- High CCM (>15)
- In Latest Snapshot
- Uncovered By Tests
- With Outstanding Issues
- With Untriaged Issues

COMPONENTS
- All In Project
- High Issue Density (>1)
- With Outstanding Issues
- With Untriaged Issues

CHECKERS
- All In Project

OWNERS
- All In Project

SNAPSHOTS

CID	Type	Impac...	Statu...	Count	First Detected	Owner	Classificatio...	Severity	Action	Component	Category
161026	Out-of-bounds access	High	New	1	06/28/13	dsteve	Unclassified	Unspecified	Undecided	Interface	Memory - corruptions
158996	Out-of-bounds access	High	New	1	06/28/13	tjg	Unclassified	Unspecified	Undecided	Interface	Memory - corruptions
158995	Out-of-bounds access	High	New	1	06/28/13	tjg	Unclassified	Unspecified	Undecided	Interface	Memory - corruptions
158994	Out-of-bounds access	High	New	1	06/28/13	tjg	Unclassified	Unspecified	Undecided	Interface	Memory - corruptions
158993	Out-of-bounds access	High	New	1	06/28/13	tjg	Unclassified	Unspecified	Undecided	Interface	Memory - corruptions
158992	Out-of-bounds access	High	New	1	06/28/13	tjg	Unclassified	Unspecified	Undecided	Interface	Memory - corruptions

1 of 37 issues selected | 1 of 1

Show ▾ /dronezone_coverity/srv/simon1/Tmp/videocore/vc4/interface/mmal/vc/mmal_server.c

```
1103        mmal_worker_port_param_get *msg = (mmal_worker_port_param_get*)vchiq_header->data;
1104        struct MMAL_CONTROL_SERVICE_T *control_service = msg->header.control_service;
1105        uint32_t component_handle = msg->component_handle;
1106        uint32_t port_handle = msg->port_handle;
1107
1108        reply.header = msg->header;
   16. Condition "vchiq_header->size == 40U /* sizeof (mmal_worker_port_param_get_old) */", taking false branch
1109        if ( vchiq_header->size == sizeof(mmal_worker_port_param_get_old) )
1110           reply.param = msg->param;
1111        else
1112        {
1113
   17. cond_at_most: Checking "msg->param.size < 392UL" implies that the value of "msg->param.size" may be up to 391 on the true branch.
   18. assignment: Assigning: "param_size" = "(msg->param.size < 392UL) ? msg->param.size : 392UL". The value of "param_size" may now be up to 391.
1114           int param_size = min(msg->param.size, sizeof(reply.param)+sizeof(reply.space));
1115           // coverity[overrun-buffer-arg] Copying param and space
1116           // coverity[overrun-buffer-val]
1117           // coverity[overrun-local]
   CID 161026 (#1 of 1): Out-of-bounds access (OVERRUN)
   19. overrun-buffer-arg: Overrunning struct type MMAL_PARAMETER_HEADER_T of 8 bytes by passing it to a function which accesses it at byte offset 390 using argument "param_size" (which evaluates to 391).
1118           memcpy(&reply.param, &msg->param, param_size);
1119        }
1120
1121        vchiq_release_message(vchiq_service, vchiq_header);
1122
1123        reply.status = mmal_server_do_port_parameter_get(server, control_service, component_handle, port_handle, &reply);
1124        if (reply.status == MMAL_SUCCESS && reply.param.id == MMAL_PARAMETER_VIDEO_DRM_PROTECT_BUFFER)
1125        {
1126           // A protected DRM buffer allocated (or deallocated) via this
1127           // parameter has bypassed the normal buffer allocation method,
1128           // and so won't be registered in the server's handle_map.
1129           // We need to register (and deregister) manually.
1130           int client_pid = control_service->client_pid;
1131           mmal_server_handle_drm_protect_buffer(server, &reply, client_pid);
1132        }
1133        send_reply(vchiq_service, &reply, MMAL_OFFSET(mmal_worker_port_param_get_reply, param) + reply.param.size);
```

161026 Out-of-bounds access

In worker: Out-of-bounds access to a buffer (CWE-119)

- Classification: Unclassified
- Severity: Unspecified
- Action: Undecided
- Ext. Reference: Type attribute text
- Owner: dsteve@ad.broadcom.com

— Last comment on 06/28/13 03:40 PM —
Found in Gerrit Coverity Scan need to know the escape text 'overrun-buffer-arg'

Apply + Next | Apply | Revert

Occurrences | History | Information

In dave_vc4_sdb_jb

Events contributing to issue:

Event	Location
17. cond_at_most	mmal_server.c:1114
18. assignment	mmal_server.c:1114
19. overrun-buffer-arg	mmal_server.c:1118

Find: overrun-buffer-arg | Next | Previous | Highlight all | Match case

++

Where Security Expertise Can Help

+ Identifying where security risks are likely to lie in their codebase
+ Writing custom rules for existing static analysis engines
+ Developing bespoke analysis tools
+ Advising on integrating automated security testing into development lifecycles

++

Conclusions

+ Static analysis can provide low-cost security checks once configured
+ ASTs and CFGs let you do all kinds of awesome things
+ Automated code analysis complements traditional manual assessments

Thanks for listening!

Questions?