

Bug Hunting with Structural Code Search

Rijnard van Tonder

grep

- ▶ Regular expression search for plain text
- ▶ Good for bug hunting

grep example on libssh2

```
ALLOC\[^[^,]*,[^;]*[*][^;]*\);
```

grep example on libssh2

```
ALLOC\[^[,]*,[^;]*[*][^;]*\);
```

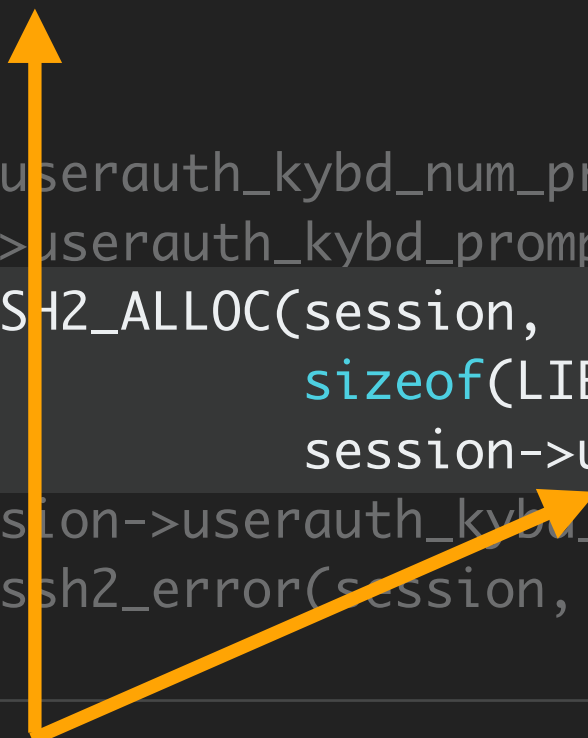


WHAT DOES IT MEAN??

grep example on libssh2

```
ALLOC\[^[,]*,[^;]*[*][^;]*\);
```

```
session->userauth_kybd_num_prompts = _libssh2_ntohu32(s);  
s += 4;  
  
if(session->userauth_kybd_num_prompts) {  
    session->userauth_kybd_prompts =  
        LIBSSH2_ALLOC(session,  
                        sizeof(LIBSSH2_USERAUTH_KBDINT_PROMPT) *  
                        session->userauth_kybd_num_prompts);  
    if (!session->userauth_kybd_prompts) {  
        _libssh2_error(session, LIBSSH2_ERROR_ALLOC,  
                        "LIBSSH2: out of memory")  
    }  
}
```



The bug: attacker controlled alloc size => integer overflow

grep example on libssh2

```
ALLOC\[^[,]*,[^;]*[*][^;]*\);
```

```
session->userauth_kybd_num_prompts = _libssh2_ntohu32(s);
s += 4;

if(session->userauth_kybd_num_prompts) {
    session->userauth_kybd_prompts =
        LIBSSH2_ALLOC(session,
            sizeof(LIBSSH2_USERAUTH_KBDINT_PROMPT) *
            session->userauth_kybd_num_prompts);
    if (!session->userauth_kybd_prompts) {
        _libssh2_error(session, LIBSSH2_ERROR_ALLOC,
```

}

grep example on libssh2

```
ALLOC\[^\,]*,[^\;]*[*][^\;]*\);
```

```
session->userauth_kybd_num_prompts = _libssh2_ntohu32(s);
s += 4;

if(session->userauth_kybd_num_prompts) {
    session->userauth_kybd_prompts =
        LIBSSH2_ALLOC(session,
            sizeof(LIBSSH2_USERAUTH_KBDINT_PROMPT) *
            session->userauth_kybd_num_prompts);
    if (!session->userauth_kybd_prompts) {
        _libssh2_error(session, LIBSSH2_ERROR_ALLOC,
```

}

grep example on libssh2

```
ALLOC\[^\,]*,[^\;]*[*][^\;]*\);
```

**METASYNTAX
(REPEAT)**

grep example on libssh2

```
ALLOC\([^\,]*,[^\;]*[*][^\;]*\);
```

**TEXT
(MULTIPLY OP)**

grep example on libssh2

```
ALLOC\([^\,]*,[^\;]*[*][^\;]*\);
```

**TEXT
(MULTIPLY OP)**

```
session->userauth_kybd_num_prompts = _libssh2_ntohu32(s);  
S  
i  
d_num_prompts) {  
    kybd_prompts =  
    session,  
    sizeof(LIBSSH2_USERAUTH_KBDINT_PROMPT) *  
    session->userauth_kybd_num_prompts);  
    if (!session->userauth_kybd_prompts) {  
        _libssh2_error(session, LIBSSH2_ERROR_ALLOC,  
            "Out of memory allocating keyboard prompts");  
    }  
}
```

grep example on libssh2

`ALLOC\([^\,]*,[^\;]*[*][^\;]*\);`

```
session->userauth_kybd_num_prompts = _libssh2_ntohu32(s);
s += 4;

if(session->userauth_kybd_num_prompts) {
    session->userauth_kybd_prompts =
        LIBSSH2_ALLOC(session,
            sizeof(LIBSSH2_USERAUTH_KBDINT_PROMPT) *
            session->userauth_kybd_num_prompts);
    if (!session->userauth_kybd_prompts) {
        _libssh2_error(session, LIBSSH2_ERROR_ALLOC,
```

grep example on libssh2

```
ALLOC\([[^,]*,[^;]*[*][^;]*\);
```

- ▶ Code structure matters

grep example on libssh2

ALLOC\([^\,]*,[^\;]*[*][^\;]*\);

- Code structure matters

grep example on libssh2

```
ALLOC\([^,]*,^;]**]^;]*\);
```

- ▶ Code structure matters
- ▶ Can we do better?

grep example on libssh2

```
ALLOC\([^\,]*,[^\;]*[*][^\;]*\);
```

comby

~~grep~~ example on libssh2

```
ALLOC\[^[,]*,[^;]*[*][^;]*\);
```


comby

~~grep~~ example on libssh2

```
ALLOC\[^[,]*,[^;]*[*][^;]*\);
```

```
ALLOC(:[1],:[2]*:[3]);
```

comby

~~grep~~ example on libssh2

```
ALLOC\[^[,]*,[^;]*[*][^;]*\);
```

```
ALLOC(:[1],:[2]*:[3]);
```

**HOLES BIND IDENTIFIERS TO
SYNTAX**

comby

~~grep~~ example on libssh2

```
ALLOC\[^[,]*,[^;]*[*][^;]*\);
```

```
ALLOC(:[1],:[2]*:[3]);
```

**CONCRETE
SYNTAX**

comby

~~grep~~ example on libssh2

```
ALLOC\[^[,]*,[^;]*[*][^;]*\);
```

```
ALLOC(:[1],:[2]*:[3]);
```

**BALANCED
DELIMITERS**

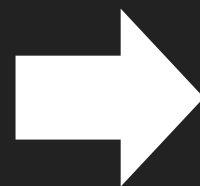
comby

~~grep~~ example on libssh2

```
ALLOC\[^[,]*,[^;]*[*][^;]*\);
```

```
ALLOC(:[1],:[2]*:[3]);
```

**BALANCED
DELIMITERS**



**NESTED CODE STRUCTURES.
LANGUAGE-AWARE.**

comby supports ~all the languages

Assembly, Bash, C/C++, C#, Clojure, CSS, Dart, Elm, Elixir, Erlang, Fortran, F#, Go, Haskell, HTML/XML, Java, Javascript, JSX, JSON, Julia, LaTeX, Lisp, Nim, OCaml, Pascal, PHP, Python, Reason, Ruby, Rust, Scala, SQL, Swift, Plain Text, TSX, Typescript

comby supports ~all the languages

Assembly, Bash, C/C++, C#, Clojure, CSS, Dart, Elm, Elixir, Erlang, Fortran, F#, Go, Haskell, HTML/XML, Java, Javascript, JSX, JSON, Julia, LaTeX, Lisp, Nim, OCaml, Pascal, PHP, Python, Reason, Ruby, Rust, Scala, SQL, Swift, Plain Text, TSX, Typescript

comby on the command line

```
rvt:libssh2/ (master*) $ with
```


comby on the Linux Kernel



comby on the Linux Kernel



Multiple NULL deref on alloc_workqueue

[CVE-2019-16230](#), [CVE-2019-16231](#), [CVE-2019-16232](#), [CVE-2019-16233](#), [CVE-2019-16234](#) • Linux Kernel • published 3 months ago • discovered by [Nico Waisman](#)

[1] [https://en.wikipedia.org/wiki/Tux_\(mascot\)#/media/File:Tux.png](https://en.wikipedia.org/wiki/Tux_(mascot)#/media/File:Tux.png)

[2] <https://securitylab.github.com/disclosures>

comby on the Linux Kernel



Multiple NULL deref on alloc_workqueue

[CVE-2019-16230](#), [CVE-2019-16231](#), [CVE-2019-16232](#), [CVE-2019-16233](#), [CVE-2019-16234](#) • Linux Kernel • published 3 months ago • discovered by [Nico Waisman](#)

There are multiple points in the Linux Kernel where `alloc_workqueue` is not getting checked for errors and as a result, a potential NULL dereference could occur.

[1] [https://en.wikipedia.org/wiki/Tux_\(mascot\)#/media/File:Tux.png](https://en.wikipedia.org/wiki/Tux_(mascot)#/media/File:Tux.png)

[2] <https://securitylab.github.com/disclosures>

[3] <https://lkml.org/lkml/2019/9/9/487>

comby on the Linux Kernel



Multiple NULL deref on alloc_workqueue

[CVE-2019-16230](#), [CVE-2019-16231](#), [CVE-2019-16232](#), [CVE-2019-16233](#), [CVE-2019-16234](#) • Linux Kernel • published 3 months ago • discovered by [Nico Waisman](#)

There are multiple points in the Linux Kernel where `alloc_workqueue` is not getting checked for errors and as a result, a potential NULL dereference could occur.

[1] [https://en.wikipedia.org/wiki/Tux_\(mascot\)#/media/File:Tux.png](https://en.wikipedia.org/wiki/Tux_(mascot)#/media/File:Tux.png)

[2] <https://securitylab.github.com/disclosures>

[3] <https://lkml.org/lkml/2019/9/9/487>

comby on the Linux Kernel



```
alloc_workqueue(:[args]);
```

There are multiple points in the Linux Kernel where `alloc_workqueue` is not getting checked for errors and as a result, a potential NULL dereference could occur.

comby on the Linux Kernel



```
alloc_workqueue(:[args]);
```

WON'T MATCH COMMENTS

comby on the Linux Kernel



```
alloc_workqueue(:[args]);
```

274 CALLS

comby on the Linux Kernel



```
alloc_workqueue(:[args]);
```

```
ppd->hfi1_wq =  
    alloc_workqueue(  
        "hfi%d_%d",  
        WQ_SYSFS | WQ_HIGHPRI | WQ_CPU_INTENSIVE |  
        WQ_MEM_RECLAIM,  
        HFI1_MAX_ACTIVE_WORKQUEUE_ENTRIES,  
        dd->unit, pidx);  
if (!ppd->hfi1_wq)  
    goto wq_error;
```

USUALLY FOLLOWED BY AN 'IF' CHECK



comby on the Linux Kernel



```
alloc_workqueue(:[args]); :[[word]]
```

```
where :[[word]] != "if"
```

RULES PLACE CONSTRAINTS ON MATCHES

comby on the Linux Kernel



```
alloc_workqueue(:[args]); :[[word]]
```

```
where :[[word]] != "if"
```

```
cgroup_destroy_wq = alloc_workqueue("cgroup_destroy", 0, 1);  
BUG_ON(!cgroup_destroy_wq);
```

SOMETIMES A DIFFERENT FLAVOR

comby on the Linux Kernel



```
alloc_workqueue(:[args]); :[[word]]
```

```
where :[[word]] != "if", :[[word]] != "BUG_ON"
```

comby on the Linux Kernel



```
alloc_workqueue(:[args]); :[[word]]
```

```
where :[[word]] != "if", :[[word]] != "BUG_ON"
```

► 38 calls left

comby on the Linux Kernel



```
alloc_workqueue(:[args]); :[[word]]
```

```
where :[[word]] != "if", :[[word]] != "BUG_ON"
```

- ▶ 38 calls left
- ▶ 1.5 minutes

comby on the Linux Kernel



```
alloc_workqueue(:[args]); :[[word]]
```

```
where :[[word]] != "if", :[[word]] != "BUG_ON"
```

► 38 calls left

► 1.5 minutes

drivers/scsi/lpfc/lpfc_init.c

```
45      /* The lpfc_wq workqueue for deferred irq use */  
46      phba->wq = alloc_workqueue("lpfc_wq", WQ_MEM_RECLAIM, 0);
```

comby on the Linux Kernel



```
alloc_workqueue(:[args]); :[[word]]
```

```
where :[[word]] != "if", :[[word]] != "BUG_ON"
```

► 38 calls left

► 1.5 minutes

drivers/staging/rtl8723bs/hal/rtl8723b_hal_init.c

```
4500 adapter->priv_checkbt_wq = alloc_workqueue("sdio_wq", 0, 0);  
4501 INIT_DELAYED_WORK(&adapter->checkbt_work, (void *)check_bt_status_work);
```

comby on the Linux Kernel



```
logger->log_workqueue = create_singlethread_workqueue("cros_usbpd_log");  
+ if (!logger->log_workqueue)  
+     return -ENOMEM;
```


comby on the Linux Kernel



```
alloc_workqueue(:[args]); :[[word]]
```



```
create_singlethread_workqueue(:[args]);
```

```
logger->log_workqueue = create_singlethread_workqueue("cros_usbpd_log");  
+ if (!logger->log_workqueue)  
+     return -ENOMEM;
```

comby on cpython



Modules/_io/winconsoleio.c

```
...
996     if (!wlen)
997         return PyErr_SetFromWindowsErr(0);
998
999     wbuf = (wchar_t*)PyMem_Malloc(wlen * sizeof(wchar_t));
1000
1001     Py_BEGIN_ALLOW_THREADS
1002     wlen = MultiByteToWideChar(CP_UTF8, 0, b->buf, len, wbuf, wlen);
1003     if (wlen) {
1004         res = WriteConsoleW(self->handle, wbuf, wlen, &n, NULL);
1005     }
1006     ...
```





comby on kubernetes



comby on kubernetes



 [kubernetes](#) / [kubernetes](#)

 [Watch](#) 3.1k [Star](#) 60.8k [Fork](#) 21.5k

[Code](#) [Issues 2,245](#) [Pull requests 1,079](#) [Actions](#) [Projects 9](#) [Security](#) [Insights](#)

Port parsing can overflow (TOB-K8S-015: Overflows when using strconv.Atoi and downcasting the result) #81121

[New issue](#)

[Open](#) cji opened this issue on Aug 7 · 19 comments

comby on kubernetes



```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

comby on kubernetes



```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

PARSED AS INT64

comby on kubernetes



```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

**CONVERTED TO INT32
=> COULD OVERFLOW**

comby on kubernetes



```
kubectl expose deployment nginx-deployment --port 4294967377 --target-port 80 service/nginx-deployment exposed
```



comby on kubernetes



kubectl expose deployment nginx-deployment --port 4294967377 --target-port 80 service/nginx-deployment exposed

```
root@k8s-1:/home/vagrant# kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.233.0.1	<none>	443/TCP	42m
nginx-deployment	ClusterIP	10.233.25.138	<none>	81/TCP	2s

Figure 13.5: The overflown port got exposed.

```
root@k8s-1:/home/vagrant# curl 10.233.25.138:81
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
```

comby on kubernetes



```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

comby on kubernetes



```
:[[v]], err := strconv.ParseInt(:[1], :[2], 64)
```

```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

comby on kubernetes



```
: [[v]], err := strconv.ParseInt(:[1], :[2], 64) :[rest]
```

```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

comby on kubernetes



```
:[[v]], err := strconv.ParseInt(:[1], :[2], 64) :[rest]
```

:[rest] stops matching
inside {...}

```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

comby on kubernetes



```
:[[v]], err := strconv.ParseInt(:[1], :[2], 64) :[rest]
```

```
where match :[rest] {  
| “int32(:[arg])” -> :[arg] == :[[v]]  
}
```

**NESTED MATCH RULE ON
INT32(...) FUNCTION**

comby on kubernetes



```
:[[v]], err := strconv.ParseInt([1], [2], 64) [rest]
```

```
where match [rest] {  
  | “int32([arg])” -> [arg] == [[v]]  
}
```

```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

comby on kubernetes



```
:[[v]], err := strconv.ParseInt(:[1], :[2], 64) :[rest]
```

```
where match :[rest] {  
| “int32(:[arg])” -> :[arg] == :[[v]]  
}
```

**ARGUMENT MUST EQUAL
PREVIOUS VARIABLE**

comby on kubernetes

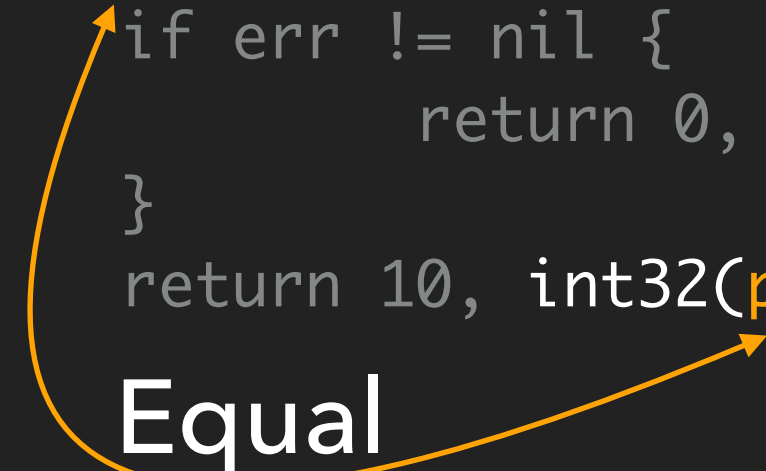


```
:  
[[v]], err := strconv.ParseInt(:[1], :[2], 64) :[rest]
```

```
where match :[rest] {  
| "int32(:[arg])" -> :[arg] == :[[v]]  
}
```

```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

Equal

An orange arrow originates from the word 'parsed' in the line 'parsed, err := strconv.ParseInt...' and points to the 'parsed' argument inside the 'int32(parsed)' function call in the line 'return 10, int32(parsed), DecimalExponent, true'.

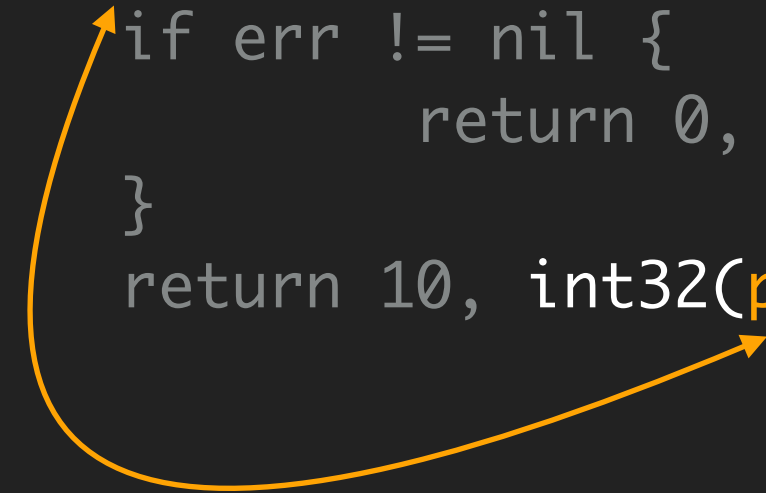
comby on kubernetes



```
:  
[[v]], err := strconv.ParseInt(:[1], :[2], 64) :[rest]
```

```
where match :[rest] {  
  | "int32(:[arg])" -> :[arg] == :[[v]]  
}
```

```
func (sh *suffixHandler) interpret(...) (...) {  
    // ...  
    parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)  
    if err != nil {  
        return 0, 0, DecimalExponent, false  
    }  
    return 10, int32(parsed), DecimalExponent, true  
}
```

A curved orange arrow originates from the variable 'parsed' in the line 'parsed, err := strconv.ParseInt...' and points to the 'parsed' argument in the line 'return 10, int32(parsed), DecimalExponent, true'.

comby on kubernetes



```
: [[v]], err := strconv.ParseInt(:[1], :[2], 64) :[rest]
```

```
where match :[rest] {  
  | "int32(:[arg])" -> :[arg] == :[[v]]  
}
```

comby on kubernetes



```
: [[v]], err := strconv.ParseInt(:[1], :[2], 64) :[rest]
```

```
where match :[rest] {  
  | "int32(:[arg])" -> :[arg] == :[[v]]  
  | "int16(:[arg])" -> :[arg] == :[[v]]  
}
```

ADDITIONAL "OR" CASES

comby on the command line

```
rvt:kubernetes/ (master) $ with-highlights comby ':[[v]], err := strconv.ParseInt(:[1], :[2], 64) :[rest]' \
'where match :[rest] { | "int32(:[arg])" -> :[arg] == :[v] }'
bck-i-search: w_
```

Structural code search: an easier way to search syntax trees



COMBY

- ▶ Effort
- ▶ Generality
- ▶ Complexity
- ▶ Precision
- ▶ Speed

Structural code search: an easier way to search syntax trees



COMBY

- ▶ Effort
- ▶ Generality
- ▶ Complexity
- ▶ Precision
- ▶ Speed

Structural code search: an easier way to search syntax trees

Assembly, Bash, C/C++, C#, Clojure, CSS, Dart, Elm, Elixir, Erlang, Fortran, F#, Go, Haskell, HTML/XML, Java, Javascript, JSX, JSON, Julia, LaTeX, Lisp, Nim, OCaml, Pascal, PHP, Python, Reason, Ruby, Rust, Scala, SQL, Swift, Plain Text, TSX, Typescript



- ▶ Effort
- ▶ Generality
- ▶ Complexity
- ▶ Precision
- ▶ Speed

Structural code search: an easier way to search syntax trees

Assembly, Bash, C/C++, C#,
Clojure, CSS, Dart, Elm, Elixir,
Erlang, Fortran, F#, Go,
Haskell, HTML/XML, Java,
Javascript, JSX, JSON, Julia,
LaTeX, ...



COMBY

- ▶ Effort
- ▶ Generality
- ▶ Complexity
- ▶ Precision
- ▶ Speed

Lightweight Multi-Language Syntax Transformation with Parser Parser Combinators

Rijnard van Tonder
School of Computer Science
Carnegie Mellon University
USA
rvt@cs.cmu.edu

Claire Le Goues
School of Computer Science
Carnegie Mellon University
USA
clegoues@cs.cmu.edu

CCS Concepts • Software and its engineering → Syntax; Translator writing systems and compiler generation; General programming languages; Domain specific languages

Abstract

grep example on libssh2

```
ALLOC\([^\],*,[^;]*[*][^;]*\);
```

- ▶ Code structure matters
- ▶ Can we do better?

comby syntax

```
ALLOC(:[1], :[2]*:[3]);
```

```
935 | LIBSSH2_ALLOC(session,
936 | list[keys].num_attrs *
937 | sizeof(libssh2_publickey_attribute));

github.com/rvantonder/libssh2-f1cfa55 > userauth.c

1504 | LIBSSH2_ALLOC(session,
1505 | sizeof(LIBSSH2_USERAUTH_KBDINT_PROMPT) *
1506 | session->userauth_kybd_num_prompts);

1518 | LIBSSH2_ALLOC(session,
1519 | sizeof(LIBSSH2_USERAUTH_KBDINT_RESPONSE) *
1520 | session->userauth_kybd_num_prompts);
```

comby on kubernetes



```
: [[v]], err := strconv.Atoi(:[1], :[2], 64) :[rest]
```

```
where match :[rest] {
| "int32(:[arg])" -> :[arg] == :[v]
}
```

```
func (sh *suffixHandler) interpret(...) (...) {
// ...
parsed, err := strconv.ParseInt(string(suffix[1:]), 10, 64)
if err != nil {
return 0, 0, DecimalExponent, false
}
return 10, int32(parsed), DecimalExponent, true
}
```

62

<https://comby.dev>



<https://github.com/comby-tools/comby>



<https://gitter.im/comby-tools/community>

