Rust: making the Internet more secure,

statically

Keith Wansbrough, Metaswitch Networks Ltd.

2018-01-24

ar music states at

metaswitch

Metaswitch Networks | © 2018 | 1

Outline

- Common vulnerabilities in software today
- The Rust programming language
- How Rust's type system ensures memory safety:
 - Ownership / Linear types
 - Lifetimes / Region types
- API safety more than just pointers
- High performance communications software
- Conclusion and further reading



Common vulnerabilities



Metaswitch Networks | © 2018 | 3

Some recent vulnerabilities in Apache httpd



Apache httpd 2.2 vulnerabilities

This page lists all security vulnerabilities fixed in released versions of Apache httpd 2.2. Each vulnerability is given a security impact rating by the Apache security team - please note that this rating may we Apache httpd the flaw is known to affect, and where a flaw has not been verified list the version with a question mark.

Please note that if a vulnerability is shown below as being fixed in a "-dev" release then this means that a fix has been applied to the development source tree and will be part of an upcoming full release.

This page is created from a database of vulnerabilities originally populated by Apache Week. Please send comments or corrections for these vulnerabilities to the Security Team.

Fixed in Apache httpd 2.2.35-dev

low: Use-after-free when using <Limit > with an unrecognized method in .htaccess ("OptionsBleed") (CVE-2017-9798)

When an unrecognized HTTP Method is given in an <Limit {method}> directive in an .htaccess file, and that .htaccess file is processed by the corresponding request, the global methods table is corrupted in the current worker process, resulting in erratic behaviour.

This behavior may be avoided by listing all unusual HTTP Methods in a global httpd.conf RegisterHttpMethod directive in httpd release 2.2.32 and later.

To permit other .htaccess directives while denying the <Limit > directive, see the AllowOverrideList directive.

Source code patch is at;

http://www.apache.org/dist/httpd/patches/apply_to_2.2.34/CVE-2017-9798-patch-2.2.patch

Note 2.2 is end-of-life, no further release with this fix is planned. Users are encouraged to migrate to 2.4.28 or later for this and other fixes.

Acknowledgements: We would like to thank Hanno Böck for reporting this issue.

Reported to security team	12th July 2017
Issue public	18th September 2017
Affects	2.2.34, 2.2.32, 2.2.31, 2.2.29, 2.2.27, 2.2.26, 2.2.25, 2.2.24, 2.2.23, 2.2.22, 2.2.21, 2.2.20, 2.2.19, 2.2.18, 2.2.17, 2.2.16, 2.2.15, 2.2.14, 2.2.13, 2.2.12, 2.2.11, 2.2.10,
	2.2.9, 2.2.8, 2.2.6, 2.2.5, 2.2.4, 2.2.3, 2.2.2, 2.2.0

Fixed in Apache httpd 2.2.34

important: Uninitialized memory reflection in mod_auth_digest (CVE-2017-9788)

The value placeholder in [Proxy-]Authorization headers of type 'Digest' was not initialized or reset before or between successive key=value assignments. by mod_auth_digest.

<u>rer</u> Providing an initial key with no '=' assignment could reflect the stale value of uninitialized pool memory used by the prior request, leading to leakage of potentially confidential information, and a segfault.

Acknowledgements: We would like to thank Robert Święcki for reporting this issue.

gi?name=CVE-2017-9798 —

Some recent vulnerabilities in Apache httpd



Apache httpd 2.2 vulnerabilities

This page lists all security vulnerabilities fixed in released versions of Apache httpd 2.2. Each vulnerability is given Please note that if a subreability is shown below as being fixed in a "dev" release then this means that a fix has be This page is created from a database of vulnerabilities originally populated by Apache Week. Please send commer

Fixed in Apache httpd 2.2.35-dev

iow: Use-after-free when using <Limit > with an unrecognized method in .htaccess ("OptionsBleed")

When an unrecognized HTTP Method is given in an <Limit (method)> directive in an .htaccess file, and that .htar This behavior may be avoided by listing all unusual HTTP Methods in a global httpd.conf RegisterHttpMethod dir To permit other .htsccess directives while denying the <Limit > directive, see the AllowOverrideList directive. Source code patch is at:

http://www.apache.org/dist/httpd/patches/apply_to_2.2.34/CVE-2017-9798-patch-2.2.patch

Note 2.2 is end-of-life, no further release with this fix is planned. Users are encouraged to migrate to 2.4.28 or lab

Acknowledgements: We would like to thank Hanno Böck for reporting this issue.

Reported to security team 12th July 2017 Issue public 18th September 2017 Affects 2 2 34, 2 2 32, 2 2 31, 2 2 29, 2 2 27, 2 2 26, 2 2 25, 2 2 24, 2 2 23, 2 2 22, 2 2 21, 2 2

Fixed in Apache httpd 2.2.34

Important: Uninitialized memory reflection in mod_auth_digest (CVE-2017-9788)

The value placeholder in (Proxy-)Authorization headers of type 'Digest' was not initialized or reset before or betwee

Providing an initial key with no '+' assignment could reflect the stale value of uninitialized pool memory used by th wiednements: We would like to thank Robert Świecki for reportion this issue

Reported to security team	28th June 2017
Issue public	11th July 2017
Update Released	11th July 2017
Affects	2 2 32 2 2 31 2 2 29 2 2 27 2 2 26 2 2 25 2 2 24 2 2 23 2 2 22 2 21 2 2 20 2 2

important: ap_get_basic_auth_pw() Authentication Bypass (CVE-2017-3167)

Use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authe Third-party module writers SHOULD use ap get basic auth components0, available in 2.2.34 and 2.4.26, inster

Acknowledgements: We would like to thank Emmanuel Drevfus for reporting this issue

Reported to security team	6th February 2017
Issue public	19th June 2017
Update Released	11th July 2017
Affects	2 2 32 2 2 31 2 2 29 2 2 27 2 2 26 2 2 25 2 2 24 2 2 23 2 2 22 2 2 21 2 2 20 2 2

important: mod_ssi Null Pointer Dereference (CVE-2017-3169)

mod_ssi may dereference a NULL pointer when third-party modules call ap_hook_process_connection() during a

-	
Reported to security team	5th December 2016
Issue public	19th June 2017
Update Released	11th July 2017

	Affects	2 2 32, 2 2 31	2 2 29	2 2 27,	2.2.26	2.2.25	2.2.24	2 2 23	2 2 22	2.2.21,	2.2.20	22
--	---------	----------------	--------	---------	--------	--------	--------	--------	--------	---------	--------	----

important: ap_find_token() Buffer Overread (CVE-2017-7668)

The HTTP strict parsing changes added in 2.2.32 and 2.4.24 introduced a bug in token list parsing, which allow A circumiation amongs: We would like to thank Janiar, Janiary (automotionmal) court for reporting this insue

Reported to security team	2% Her 2017
Issue public	19th June 2017
Update Released	11th July 2017
Affects	2.2.32

Important: mod_mime Buffer Overread (CVE-2017-7679)

mod_mime can read one byte past the end of a buffer when sending a malicious Content-Type response header.

Acknowledgements: We would like to thank ChenQin and Hanno Böck for reporting this issue

Reported to security team	15th November 2015
issue public	19th June 2017

Reported to security team	15th November 2015
Issue public	19th June 2017
Lindate Released	49% July 2017

. 2 2 32, 2 2 31, 2 2 29, 2 2 27, 2 2 26, 2 2 25, 2 2 24, 2 2 23, 2 2 22, 2 2 21, 2 2 20, 2 2 19, 2 2 18, 2 2 17, 2 2 Affects

Fixed in Apache httpd 2.2.32

Important: Apache HTTP Request Parsing Whitespace Defects (CVE-2016-8743)

Apache HTTP Server, prior to release 2.4.25 (2.2.32), accepted a broad pattern of unusual whitespace patterns from the user-agent, Inc header field name would be honored as whitespace, and a bare CR in the request header field value was retained the input headers are

RFC7230 Section 3.5 calls out some of these whitespace exceptions, and section 3.2.3 eliminated and clarified the role of implied wh character whatsoever. Section 3.2.4 explicitly disallowed any whitespace from the request header field prior to the " character, while Se

These defects represent a security concern when httpd is participating in any chain of proxies or interacting with back-end application s proxy agent, in a sequence of two requests, this results in request A to the first proxy being interpreted as requests A + A by the backeng

These defects are addressed with the release of Apache HTTP Server 2.4.25 and coordinated by a new directive;

HttpProtocolOptions Strict

which is the default behavior of 2.4.25 and later. By togoling from 'Strict' behavior to 'Unsafe' behavior, some of the restrictions may be reallow other RFC requirements to not be enforced, such as exactly two SP characters in the request line.

Acknowledgements: We would like to thank David Dennerline at IBM Security's X-Force Researchers as well as Régis Leroy for each rec

Reported to security team	1 10th February 2016
Issue public	20th December 2016
Update Released	13th January 2017
Affects	2 2 31, 2 2 29, 2 2 27, 2 2 26, 2 2 25, 2 2 24, 2 2 23, 2 2 22, 2 2 21, 2 2 20, 2 2 19, 2 2 18, 2 2 17, 2 2 16, 2 2

n/a: HTTP_PROXY environment variable "httpoxy" mitigation (CVE-2016-5387)

HTTP_PROXY is a well-defined environment variable in a CGI process, which collided with a number of libraries which falled to avoid c

This workaround and patch are documented in the ASF Advisory at https://www.apache.org/security/asf-httpoxy-response.tx

Acknowledgements: We would like to thank Dominic Scheirlinck and Scott Geary of Vend for reporting and proposing a fix for this issue.

Reported to security team	2nd July 2016
Issue public	18th July 2016
Update Released	18th July 2016
Affects	2 2 31, 2 2 29, 2 2 27, 2 2 26, 2 2 25, 2 2 24, 2 2 23, 2 2 22, 2 2 21, 2 2 20, 2 2 19, 2 2 18, 2 2 17, 2 2 16, 2 2

Fixed in Apache httpd 2.2.31

low: HTTP request smuggling attack against chunked request parser (CVE-2015-3183)

An HTTP request smuggling attack was possible due to a bug in parsing of chunked requests. A malicious client could force the server Acknowledgements: This issue was reported by Régis Leroy.

Reported to security team	4th April 2015
Issue public	9th June 2015
Update Released	16th July 2015
Affects	2 2 29, 2 2 27, 2 2 26, 2 2 25, 2 2 24, 2 2 23, 2 2 22, 2 2 21, 2 2 20, 2 2 19, 2 2 18, 2 2 17, 2 2 16, 2 2 15, 2 2

Fixed in Apache httpd 2.2.29

important: mod_cgid denial of service (CVE-2014-0231)

A flaw was found in mod_cgld. If a server using mod_cgld hosted CGI scripts which did not consume standard input, a remote attacker

Reported to security team	16th June 2014
issue public	14th July 2014
Update Released	3rd September 2014
Affects	2 2 27, 2 2 26, 2 2 25, 2 2 24, 2 2 23, 2 2 22, 2 2 21, 2 2 20, 2 2 19, 2 2 18, 2 2 17, 2 2 16, 2 2 15, 2 2 14, 2 2

low: HTTP Trailers processing bypass (CVE-2013-5704)

HTTP trailers could be used to replace HTTP headers late during request processing, potentially undoing or otherwise confusing modu

This fix adds the "MergeTrailers" directive to restore legacy behavior.

Acknowledgements: This issue was reported by Martin Holst Swende.

Reported to security team 6th September 2013 19th October 2013 issue public

Update Release 3rd September 201 2 2 27, 2 2 26, 2 2 25, 2 2 24, 2 2 23, 2 2 22, 2 2 21, 2 20 22

moderate: mod_defiate denial of service (CVE-2014-0118)

A resource consumption flaw was found in mod_deflate. If request body decompression was configured (using the 'DEFLATE' input filte

Metaswitch Networks | © 2018 | 5

Acknowledgements: This issue was reported by Glancario Pellegring and Davide Balzarotti

Reported to security team 19th February 2014

Affects

moderate: mod status buffer overflow (CVE-2014-0226)

A race condition was found in modi status. An attacker able to access a public server status page on a server using a threaded I

Acknowledgements: This issue was reported by Marek Kroemeke, AKAT-1 and 22733db72ab3ed94b5f8a1ffcde850251fe6f466 vi

Reported to security team	30th May 2014
Issue public	14th July 2014
Update Released	3rd September 2014
Affects	

Fixed in Apache httpd 2.2.27

low: mod_log_config crash (CVE-2014-0098)

A flaw was found in mod_log_config. A remote attacker could send a specific truncated cookie causing a crash. This crash wou

Acknowledgements: This issue was reported by Rainer M Canavan

Reported to security team	25th February 2014
Issue public	17th March 2014
Update Released	26th March 2014
Affects	2 2 26, 2 2 25, 2 2 24, 2 2 23, 2 2 22, 2 2 21, 2 2 20, 2 2 19, 2 2 18, 2 2 17, 2 2 16, 2 2 15, 2 2 14, 2 2

moderate: mod dav crash (CVE-2013-6438)

XXL parsing code in modi day incorrectly calculates the end of the string when removing leading spaces and places a NUL char

Acknowledgements: This issue was reported by Ning Zhang & Amin Tora of Neustar

Reported to security team 10th December 2013 17th March 2014 Issue public 26th March 2014 Update Released 2 2 26, 2 2 25, 2 2 24, 2 2 23, 2 2 22, 2 2 21, 2 2 20, 2 2 19, 2 2 18, 2 2 17, 2 2 16, 2 2 15, 2 2 14, 2 2 Affects

Fixed in Apache httpd 2.2.25

low: mod_rewrite log escape filtering (CVE-2013-1862)

mod_rewrite does not filter terminal escape sequences from logs, which could make it easier for attackers to insert those seque

Acknowledgements: This issue was reported by Ramiro Molina

Reported to security team	13th March 2013
Issue public	19th April 2013
Update Released	22nd July 2013
Affects	2 2 23, 2 2 22, 2 2 21, 2 2 20, 2 2 19, 2 2 18, 2 2 17, 2 2 16, 2 2 15, 2 2 14, 2 2 13, 2 2 12, 2 2 11, 2 2

. . .

metaswitch

moderate: mod dav crash (CVE-2013-1896)

Sending a MERGE request against a URI handled by modi day syn with the source href (sent as part of the request body as XU/L

Acknowledgements: This issue was reported by Ben Reser

Reported to security team	7th March 2013
Issue public	23rd May 2013
Update Released	22nd July 2013
Affects	2 2 23, 2 2 22, 2 2 21, 2 2 20, 2 2 19, 2 2 18, 2 2 17, 2 2 16, 2 2 15, 2 2 14, 2 2 13, 2 2 12, 2 2 11, 2 2

Fixed in Apache httpd 2.2.24

low: X\$\$ due to unescaped hostnames (CVE-2012-3499)

Various XSS flaws due to unescaped hostnames and URIs HTML output in mod_info, mod_status, mod_imagemap, mod_idap,

Acknowledgements: This issue was reported by Niels Heinen of Google

Reported to security team	11th July 2012
issue public	18th February 2013
Update Released	25th February 2013
Affects	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

moderate: X\$\$ in mod_proxy_balancer (CVE-2012-4558)

A XSS flaw affected the modi proxy balancer manager interface

Acknowledgements: This issue was reported by Niels Heinen of Google

Reported to security team 7th October 2012

8th February 201: itv/vulnerabilities 22.html See In Cultate (eless

Possible XSS for sites which use mod_negotiation and allow untrusted uploads to locations which have MultiViews enabled

low: X\$\$ in mod_negotiation when untrusted uploads are supported (CVE-2012-2687)

Fixed in Apache httpd 2.2.23

CVE-2010-0425 (found by senseofsecurity.com.au)

- <u>CVE</u>: "modules/arch/win32/mod_isapi.c in mod_isapi in the Apache HTTP Server 2.0.37 through 2.0.63, 2.2.0 through 2.2.14, and 2.3.x before 2.3.7, when running on Windows, does not ensure that request processing is complete before calling isapi_unload for an ISAPI .dll module, which allows remote attackers to execute arbitrary code via unspecified vectors related to a crafted request, a reset packet, and "orphaned callback pointers.""
- Initial report: "By sending a specially crafted request followed by a reset packet it is possible to trigger a vulnerability in Apache mod_isapi that will unload the target ISAPI module from memory. However function pointers still remain in memory and are called when published ISAPI functions are referenced. This results in a dangling pointer vulnerability. Successful exploitation results in the execution of arbitrary code with SYSTEM privileges."



Common Vulnerabilities and Exposures The Standard for Information Security Vulnerability Names



Dangling pointer vulnerability in Apache httpd (CVE-2010-0425)

- Load a module to handle a request.
- Install a callback to invoke the module.
- Request fails.
- Unload the module.
- Trigger the callback (e.g., by a subsequent request which reloads the module).
- Callback tries to call the module at the old address,
 - but instead calls the new contents of that address which are now part of the second request.
- Request has been carefully crafted to put x86 instructions at that location.
- Attacker's code is executed.



The fix

Do not unload an isapi .dll module until the request processing is completed, avoiding orphaned callback pointers. Submitted by: Brett Gervasoni <brettg senseofsecurity.com>, trawick Reviewed by: trawick, wrowe

```
-- httpd/httpd/trunk/modules/arch/win32/mod isapi.c
                                                          2010/03/02 04:18:45
                                                                                       917869
+++ httpd/httpd/trunk/modules/arch/win32/mod isapi.c
                                                          2010/03/02 04:30:33
                                                                                       917870
@@ -1503,7 +1503,6 @@
     /* Set up the callbacks */
     cid->ecb->ReadClient = ReadClient; ... // for example; others omitted for space on slide
     /* Set up client input */
     res = ap setup client block(r, REQUEST CHUNKED ERROR);
     if (res) {
         isapi unload(isa, 0);
         return res;
@@ -1534,7 +1533,6 @@
         if (res < 0) {
             isapi unload(isa, 0);
             return HTTP INTERNAL SERVER ERROR;
```

<u>http://svn.apache.org/viewvc?view=revision&revision=917870</u>



The fix

Do not unload an isapi .dll module until the request process Submitted by: Brett Gervasoni <brettg senseofsecurity.com>, Reviewed by: trawick, wrowe

```
/* Set up the callbacks */
```

cid->ecb->ReadClient = ReadClient; ... // for example; others omnated for space on snae

```
/* Set up client input */
  res = ap_setup_client_block(r, REQUEST_CHUNKED_ERROR);
  if (res) {
        isapi_unload(isa, 0);
        return res;
     }
@@ -1534,7 +1533,6 @@
        }
        if (res < 0) {
            isapi_unload(isa, 0);
            return HTTP_INTERNAL_SERVER_ERROR;
        }
</pre>
```

<u>http://svn.apache.org/viewvc?view=revision&revision=917870</u>

- Need two rules to fix:
 - Once something is freed, it can't be used (*ownership*).
 - An object must be valid as long as any reference to it exists (*lifetime*).



```
metaswitch
```





Metaswitch Networks | © 2018 | 10

The Rust programming language

- A modern programming language (2009), originated and sponsored by Mozilla Labs (makers of Firefox)
- Focus on performance and safety
 - "Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety."
 <u>rust-lang.org</u>
 - "Rust is a systems language pursuing the trifecta: safe, concurrent, and fast" <u>This Week in Rust</u>

```
fn main() {
   let greetings = ["Hello", "Ciao", "こんにちは",
                   "안녕하세요", "Cześć", "01á"];
   for (num, greeting) in greetings.iter().enumerate() {
       print!("{} : ", greeting);
       match num {
          0 => println!("This code is editable and runnable!"),
           1 => println!("Questo codice è modificabile ed eseguibile!"),
           2 => println!("このコードは編集して実行出来ます!"),
           3 => println!("여기에서 코드를 수정하고 실행할 수 있습니다!"),
          4 => println!("Ten kod można edytować oraz uruchomić!"),
           5 => println!("Este código é editável e executável!"),
            => {},
                                                                            metaswitch
```

Fast, concurrent, safe

• Fast

- Compiles to native machine code. Competes with C/C++ for raw speed.
- No runtime, no scheduler, no startup.
- No garbage collector predictable performance.
- Zero-cost abstractions.

Concurrent

- Multi-threaded or coroutine-based
- Can use multiple cores
- Protection from data races
- Threadsafe toolkit in standard library



Other languages:

C/C++:

- Fast: yes
- Concurrent: yes
- Safe: no

Java/Scala:

- Fast: sort-of
- Concurrent: yes
- Safe: sort-of



- Safe
 - Memory safety: no double frees, no NULL pointers, no dangling pointers, no scribblers
 - API safety: compile-time API checking, no "unexpected behaviour"
- It's also pleasant to use: modern features, good IDEs, clear error messages, etc.

```
fn foo() {
    let v = vec![1, 2, 3];
}
```



let v = vec![1, 2, 3];

let $v^2 = v$;

```
println!("v[0] is: {}", v[0]);
```





let v = vec![1, 2, 3];

let $v^2 = v$;

```
println!("v[0] is: {}", v[0]);
```

fn take(v: Vec<i32>) {
 // What happens here isn't important.
}

let v = vec![1, 2, 3];

take(v);

println!("v[0] is: {}", v[0]);



let v = vec![1, 2, 3];

let $v_2 = v_3$;

```
println!("v[0] is: {}", v[0]);
```

error: use of moved value: `v` println!("v[0] is: {}", v[0]); Λ

fn take(v: Vec<i32>) { // What happens here isn't important.

let v = vec![1, 2, 3];

take(v);

}

println!("v[0] is: {}", v[0]);

error: use of moved value: `v` println!("v[0] is: {}", v[0]); Λ



Borrowing

```
fn main() {
    // This borrows an immutable reference.
    fn sum vec(v: &Vec<i32>) -> i32 {
       v.iter().fold(0, |a, \&b| a + b)
    }
      Borrow two vectors and sum them.
    // This kind of borrowing does not allow mutation through the borrowed reference.
    fn foo(v1: &Vec<i32>, v2: &Vec<i32>) -> i32 {
       // Do stuff with v1 and v2.
       let s1 = sum_vec(v1);
       let s_2 = s_1 v_2;
       // Return the answer.
       s1 + s2
    }
    let v1 = vec![1, 2, 3];
    let v_2 = vec![4, 5, 6];
    let answer = foo(\&v1, \&v2);
    println!("{}", answer);
```

Borrowing

```
fn main() {
    // This borrows an immutable reference.
    fn sum vec(v: &Vec<i32>) -> i32 {
        v.iter().fold(0, |a, \&b| a + b)
      Borrow two vectors and sum them.
    // This kind of borrowing does not allow mutation through the borrowed reference.
    fn foo(v1: &Vec<i32>, v2: &Vec<i32>) -> i32 {
        // Do stuff with v1 and v2.
        let s1 = sum_vec(v1);
        let s_2 = s_v(v_2);
                                                 Can't mutate via v1 or v2.
        // Return the answer.
        s1 + s2
                                                E.g., v1.push(42); yields:
    }
                                                 error: cannot borrow
                                          immutable borrowed content `*v1`
    let v1 = vec![1, 2, 3];
                                                      as mutable
    let v2 = vec![4, 5, 6];
    let answer = foo(\&v1, \&v2);
    println!("{}", answer);
```

```
let mut x = 5;
{
    let y = &mut x;
    *y += 1;
}
println!("{}", x);
```



```
let mut x = 5;
{
    let y = &mut x;
    *y += 1;
}
println!("{}", x);
```

6

let mut x = 5; let y = &mut x; *y += 1; println!("{}", x);

let mut x = 5; let y = &mut x; *y += 1; println!("{}", x); error: cannot borrow `x` as immutable because it is also borrowed as mutable println!("{}", x); Λ note: previous borrow of `x` occurs here; the mutable borrow prevents subsequent moves, borrows, or modification of `x` until the borrow ends let y = &mut x; Λ note: previous borrow ends here fn main() {

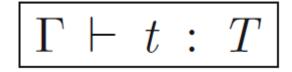
Borrowing rules in Rust

- Any borrow must last for a scope no greater than that of the owner.
- Each resource may have **either one or the other** of these two kinds of borrows, but not both at the same time:
 - One or more references (&T) to a resource.
 - Exactly one mutable reference (&mut T) to a resource.
- These rules are sufficient to make data races impossible in Rust!

There is a 'data race' when two or more pointers access the same memory location at the same time, where at least one of them is writing, and the operations are not synchronized.

- Also prevents things like:
 - Iterator invalidation (mutating a collection while you're iterating over it).
 - Use after free





$$\frac{\Gamma_{1} \vdash e_{1}: T_{2} \rightarrow T_{1} \qquad \Gamma_{2} \vdash e_{2}: T_{2}}{\Gamma_{1}, \Gamma_{2} \vdash e_{1}e_{2}: T_{1}} \qquad \text{(App)}$$

Girard, Linear Logic, 1987 Wadler, A taste of linear logic, 1993 Jung et al, RustBelt, 2018

$$\Gamma \vdash t : T$$

$$\frac{\Gamma_{1} \vdash e_{1}: T_{2} \rightarrow T_{1} \qquad \Gamma_{2} \vdash e_{2}: T_{2}}{\Gamma_{1}, \Gamma_{2} \vdash e_{1}e_{2}: T_{1}} \qquad \text{(App)}$$

$$\begin{array}{c} \hline cake : T \vdash cake : T \\ cake : T \vdash have(cake) : () \\ \hline cake : T, bread : T \vdash have(cake); eat(bread) : () \\ \hline cake : T, bread : T \vdash have(cake); eat(bread) : () \\ \hline \end{array}$$

Girard, Linear Logic, 1987 Wadler, A taste of linear logic, 1993 Jung et al, RustBelt, 2018

 $\Gamma \vdash t : T$

$$\frac{\Gamma_{1} \vdash e_{1}: T_{2} \rightarrow T_{1} \qquad \Gamma_{2} \vdash e_{2}: T_{2}}{\Gamma_{1}, \Gamma_{2} \vdash e_{1}e_{2}: T_{1}} \qquad \text{(App)}$$

$$\begin{array}{c} \hline cake : T \vdash cake : T \\ cake : T \vdash have(cake) : () \\ \hline cake : T, bread : T \vdash have(cake); eat(cake) : ? \end{array}$$

Girard, Linear Logic, 1987 Wadler, A taste of linear logic, 1993 Jung et al, RustBelt, 2018

 $\Gamma \vdash t : T$

$$\frac{\Gamma_1 \vdash e_1: T_2 \rightarrow T_1 \quad \Gamma_2 \vdash e_2: T_2}{\Gamma_1, \Gamma_2 \vdash e_1 e_2: T_1} \quad \text{(App)}$$

$cake : T \vdash cake : T$		$bread$: $T \vdash cake$: error!
$cake : T \vdash have(cake) : ()$		$bread$: $T \vdash eat(cake)$: error!
cake : T, bread :	$T \vdash$	have(cake); eat(cake) : error!

Girard, Linear Logic, 1987 Wadler, A taste of linear logic, 1993 Jung et al, RustBelt, 2018

Lifetimes

Metaswitch Networks | © 2018 | 29

Lifetimes in Rust

- Lending out a reference to a resource owned by someone else:
 - I acquire a handle to some kind of resource.
 - I lend you a reference to the resource.
 - I decide I'm done with the resource, and deallocate it, while you still have your reference.
 - You decide to use the resource. *BOOM dangling pointer / use after free!*

let r; {	<pre>// Introduce reference: `r`.</pre>
let i = 1; r = &i	<pre>// Introduce scoped value: `i`. // Store reference of `i` in `r`.</pre>
}	<pre>// `i` goes out of scope and is dropped.</pre>
<pre>println!("{}", r);</pre>	<pre>// `r` still refers to `i`.</pre>

• (Rust uses explicit memory management – not garbage collection.)



Lifetimes

```
fn skip_prefix(line: &str, prefix: &str) -> &str {
   // ...
let line = "lang:en=Hello World!";
let lang = "en";
let v;
   let p = format!("lang:{}=", lang); // -+ `p` comes into scope.
   v = skip_prefix(line, p.as_str()); // |
                                       // -+ `p` goes out of scope.
println!("{}", v);
```

Lifetimes

```
fn skip_prefix<'a, 'b>(line: &'a str, prefix: &'b str) -> &'a str {
   // ...
let line = "lang:en=Hello World!";
let lang = "en";
let v;
    let p = format!("lang:{}=", lang); // -+ `p` comes into scope.
    v = skip_prefix(line, p.as_str()); // |
                                       // -+ `p` goes out of scope.
println!("{}", v);
```

More lifetimes

```
struct Foo<'a> {
    x: &'a i32,
}
fn show() {
    let y = &5; // This is the same as `let _y = 5; let y = &_y;`.
    let f = Foo { x: y };
    println!("{}", f.x);
}
```

let x: &'static str = "Hello, world.";

• Inference and elision



Region types = lifetimes

```
letregion \rho_4, \rho_5 in
  let f =
    letregion \rho_6 in
    let x = (2 at \rho_2, 3 at \rho_6) at \rho_4;
        (|y| (x.0,y) at \rho_1) at \rho_5
    at \rho_5;
    f(5 at \rho_3)
```

- Annotate program and types with regions in which allocation occurs.
- Constrain which expressions can use which regions.

Tofte & Talpin, Region-Based Memory Management, 1994 Morrisett et al, Region-Based Memory Management in Cyclone, 2001 Jung et al, RustBelt, 2018

Metaswitch Networks | © 2018 | 34



In practice

- The type system is picky, but it ensures that Rust code cannot access a dangling pointer, or change a value under the feet of another caller.
- These guarantees are zero-cost: all the work is done at compile time.
- Learning curve of using a type system:
 - "Many new users to Rust experience something we like to call 'fighting with the borrow checker', where the Rust compiler refuses to compile a program that the author thinks is valid. This often happens because the programmer's mental model of how ownership should work doesn't match the actual rules that Rust implements. You probably will experience similar things at first. There is good news, however: more experienced Rust developers report that once they work with the rules of the ownership system for a period of time, they fight the borrow checker less and less."



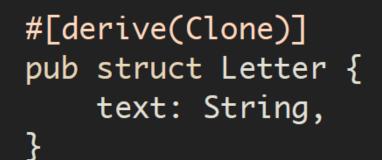
More than just pointers



Metaswitch Networks | © 2018 | 36

Resources







pub struct Envelope {
 letter: Option<Letter>,



pub struct PickupLorryHandle {
 done: bool,
 // references to lorry's resources

From: *A hammer you can only hold by the handle* (*video, slides*) <u>Andrea Lattuada</u> (ETH Zürich), RustFest 2017

A dangerous API 1



}

pub struct Envelope {
 letter: Option<Letter>,

```
impl Envelope {
    pub fn wrap(&mut self, letter: &Letter) {
        self.letter = Some(letter.clone());
    }
}
pub fn buy_prestamped_envelope() -> Envelope {
    Envelope { letter: None }
}
```

A dangerous API 2

```
pub struct PickupLorryHandle {
                 done: bool,
                 // references to lorry's resources
ワ
             }
impl PickupLorryHandle {
    pub fn pickup(&mut self, envelope: &Envelope) {
        /* give letter */
    pub fn done(&mut self) {
        self.done = true; println!("sent");
    }
}
pub fn order_pickup() -> PickupLorryHandle {
    PickupLorryHandle { done: false, /* other handles */ }
}
```

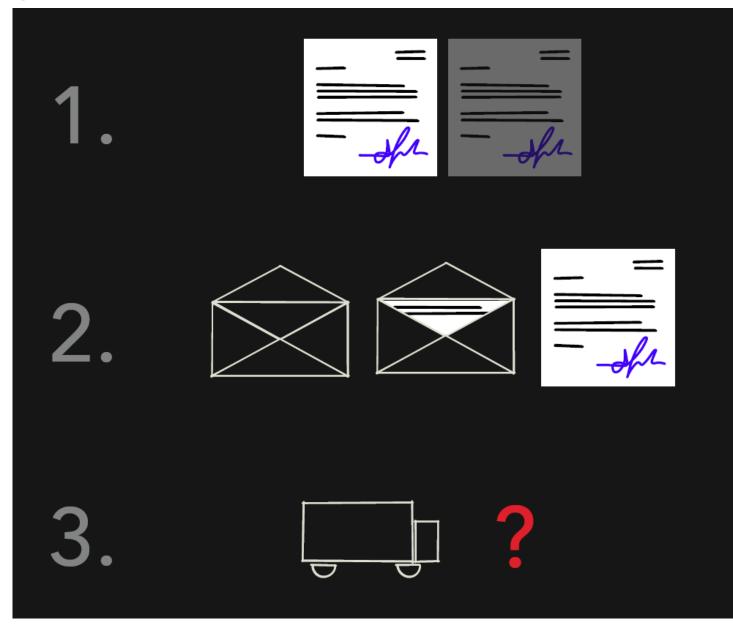
Using the dangerous API

```
// in a separate module
fn main() {
    let rustfest_letter = Letter::new(String::from("Dear RustFest"));
    let mut rustfest_envelope = buy_prestamped_envelope();
    rustfest_envelope.wrap(&rustfest_letter);
    let mut lorry = order_pickup();
    lorry.pickup(&rustfest_envelope);
    lorry.done();
```

}



What can go wrong?





Constraining the dangerous API with ownership types

```
pub struct EmptyEnvelope { }
pub struct ClosedEnvelope {
    letter: Letter,
impl EmptyEnvelope {
   pub fn wrap(self, letter: Letter) -> ClosedEnvelope {
        ClosedEnvelope { letter: letter }
impl PickupLorryHandle {
   pub fn pickup(&mut self, envelope: ClosedEnvelope) {
        /* give letter */
pub fn buy_prestamped_envelope() -> EmptyEnvelope { EmptyEnvelope { } }
```



Attempting to put the same letter into two envelopes

```
fn main() {
    let rustfest_letter = Letter::new(String::from("Dear RustFest"));
    let mut envelopes = vec![
        buy_prestamped_envelope(), buy_prestamped_envelope()];
    let mut lorry = order_pickup();
    for e in envelopes.iter_mut() {
        e.wrap(rustfest_letter);
    }
}
```

```
lorry.pickup(&e);
}
lorry.done();
```

```
metaswitch
```

Attempting to put the same letter into two envelopes

error[E0382]: use of moved value: `rustfest_letter`

```
fn main() {
   let rustfest_letter = Letter::new(String::from("Dear RustFest"));
   let mut envelopes = vec![
       buy_prestamped_envelope(), buy_prestamped_envelope()];
   let mut lorry = order_pickup();
   for e in envelopes.iter_mut() {
       e.wrap(rustfest_letter);
              ^^^^^ value moved here in previous iteration
       lorry.pickup(&e); of loop
   lorry.done();
```



High-performance comms software



Metaswitch Networks

 "Metaswitch is the world's leading cloud native communications software company. We develop commercial and open-source software solutions that are constructively disrupting the way that service providers build, scale, innovate and account for communication services."

Write and sell telecoms/networking software
Some mobile and web apps...

...but mainly back-end servers to make it all work

- Work with customers to put it all together
- Support it
- Complexity, scale, reliability,....

Metaswitch Networks

- Shift to microservices architecture
- Great opportunity to re-select a modern language to use henceforth
 Existing code: C, C++, Java, Scala, Python, Perl
 Productive, but mixed experiences with all of them
 Needed: safety, flexibility, FFI, tooling, ecosystem, and fun!
 Rust ticked all the boxes.
- We're well along the path, gaining experience, and loving it.

Conclusion



Conclusion

- Software engineering has a problem
 - Common vulnerabilities
- PL type systems offer a solution
 - Linear types and region types
- Rust shows that it works in practice
 - Fast, concurrent, safe pick three
 - Easily adopted
 - Used in production

Further reading:

- Francois Pottier, A Linear Bestiary
- <u>Ralf Jung et al, RustBelt: Securing the Foundations</u> of the Rust Programming Language (POPL 2018)
- https://www.rust-lang.org/
- Wikipedia on Rust
- <u>Rust Book (1st ed) bibliography types</u>
- <u>The Rust Book (1st ed): chapter on ownership</u>
- <u>Metaswitch blog</u> (e.g., <u>1</u>, <u>2</u>)

Contact me: Keith.Wansbrough@metaswitch.com

@kw217

