Rust: using linear and region types to make the Internet more secure

Keith Wansbrough, Metaswitch Networks Ltd.

2017-11-20

metaswitch

Outline

- Common vulnerabilities in software today
- Linear types
- Region types
- The Rust programming language
- How Rust's type system ensures memory and API safety
- High performance communications software
- Conclusion and further reading

Common vulnerabilities

Some recent vulnerabilities in Apache httpd

APACHE HTTP SERVER PROJECT

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.

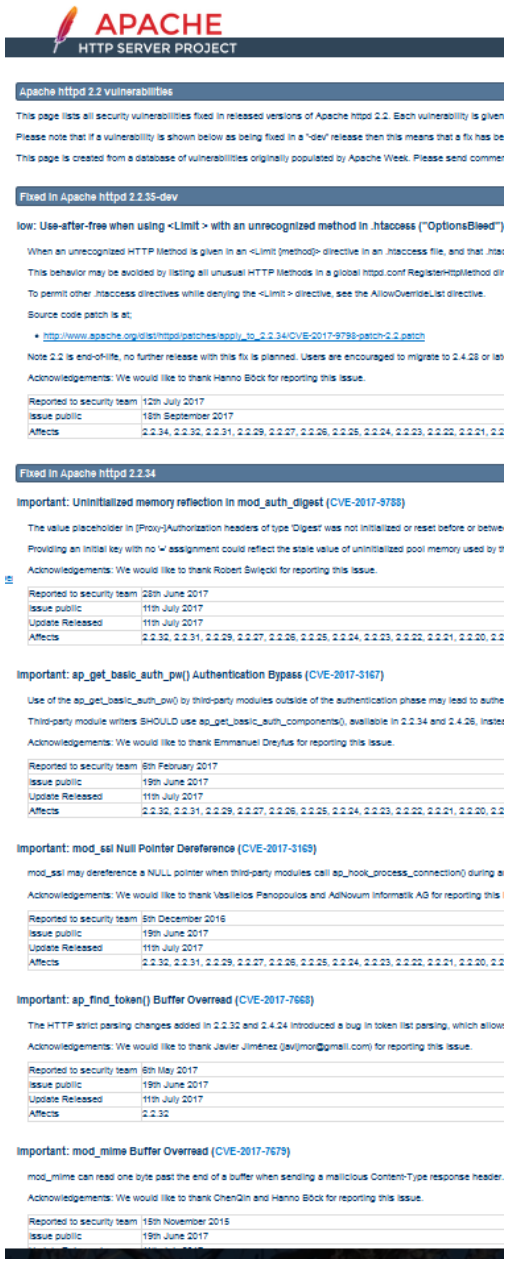
Providing an initial key with no '=' assignment could reflect the stale value of uninitialized pool memory used by the prior request, leading to leakage of potentially confidential information, and a segfault.

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

gi?name=CVE-2017-9798

metaswitch

Some recent vulnerabilities in Apache httpd



APACHE HTTP SERVER PROJECT

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 vulnerability 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 comme

Fixed in Apache httpd 2.2.35-dev

low: 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 .hta

This behavior may be avoided by listing all unusual HTTP Methods in a global httpd.conf RegisterHttpMethod dir

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 lat

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 betwe

Providing an initial key with no '=' assignment could reflect the stale value of uninitialized pool memory used by t

Acknowledgements: We would like to thank Robert Święcki for reporting 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, 2.2.21, 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_components(), available in 2.2.34 and 2.4.26, instea

Acknowledgements: We would like to thank Emmanuel Dreyfus 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_ssl Null Pointer Dereference (CVE-2017-3169)

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

Acknowledgements: We would like to thank Vasileios Panopoulos and AdNovum Informatik AG for reporting this i

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, 2.2

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 allows

Acknowledgements: We would like to thank Javier Jiménez (javijmor@gmail.com) for reporting this issue.

Reported to security team	6th May 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
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, 2.2.19, 2.2.18, 2.2.17, 2.2

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, incl
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 whi
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 backend

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 toggling from 'Strict' behavior to 'Unsafe' behavior, some of the restrictions may be re
allow 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 rep

Reported to security team	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 failed to avoid c

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

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 serve

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_cgid. If a server using mod_cgid hosted CGI scripts which did not consume standard input, a remote attacker

Acknowledgements: This issue was reported by Rainer Jung of the ASF

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
Issue public	19th October 2013
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

moderate: mod_deflate 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 filt

Acknowledgements: This issue was reported by Giancarlo Pellegrino and Davide Balzarotti

Reported to security team	19th February 2014
Issue public	14th July 2014

moderate: mod_status buffer overflow (CVE-2014-0226)

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

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	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 2.2.27

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)

XML parsing code in mod_dav incorrectly calculates the end of the string when removing leading spaces and places a NUL cha

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

Reported to security team	10th December 2013
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

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

moderate: mod_dav crash (CVE-2013-1896)

Sending a MERGE request against a URI handled by mod_dav_svn with the source href (sent as part of the request body as XML

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: XSS 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_ldap,

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	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

moderate: XSS in mod_proxy_balancer (CVE-2012-4558)

A XSS flaw affected the mod_proxy_balancer manager interface.

Acknowledgements: This issue was reported by Niels Heinen of Google

Reported to security team	7th October 2012
Issue public	18th February 2013

Fixed in Apache httpd 2.2.23

low: XSS in mod_negotiation when untrusted uploads are supported (CVE-2012-2687)

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

...

http://httpd.apache.org/security/vulnerabilities_22.html

metaswitch

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

- CVE: “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

metaswitch

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 proces
Submitted by: Brett Gervasoni <brettg senseofsecurity.com>,
Reviewed by: trawick, wrowe

-- httpd/httpd/trunk/modules/arch/win32/mod_isapi.c     2010
+++ httpd/httpd/trunk/modules/arch/win32/mod_isapi.c    2010
@@ -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;
         }
```

- 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***).

- http://svn.apache.org/viewvc?view=revision&revision=917870

Linear types

A simple type system

$$\frac{\Gamma \vdash e_1 : T_1 \quad \Gamma, x : T_1 \vdash e_2 : T_2}{\Gamma \vdash \text{let } x = e_1 \text{ in } e_2 : T_2} \text{ (Let)}$$

$$\frac{}{\Gamma, x : T \vdash x : T} \text{ (Var)}$$

$$\frac{\Gamma \vdash e_1 : T_2 \to T_1 \quad \Gamma \vdash e_2 : T_2}{\Gamma \vdash e_1\, e_2 : T_1} \text{ (App)}$$

- `let x = bake_a_cake();`
- `eat(x);`
- `have(x);`
- `eat(x)`

- Syntactic sugar:
  - `e1;e2`
    `==  let _ = e1 in e2`
  - `let x = e1; e2`
    `==  let x = e1 in e2`
  - `e1(e2)`
    `== e1 e2`

Fixing the type rules – a linear type system

$$\frac{\Gamma_1 \vdash e_1 : T_1 \qquad \Gamma_2, x : T_1 \vdash e_2 : T_2}{\Gamma_1, \Gamma_2 \vdash \text{let } x = e_1 \text{ in } e_2 : T_2} \text{ (Let)}$$

$$\frac{}{x : T \vdash x : T} \text{ (Var)}$$

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

- ```
  let x = bake_a_cake();    // 15
  eat(x);                   // 16
  have(x);                  // 17
  eat(x)                    // 18
  ```

```
error[E0382]: use of moved value: `x`
  --> src/main.rs:17:10
   |
16 |     eat(x);
   |         - value moved here
17 |     have(x);
   |          ^ value used here after move
   |
   = note: move occurs because `x` has type
`std::string::String`, which does not implement the
`Copy` trait
```
```

Region types

Two programs (with block scoped allocation)

```
let f = {let x = (2,3);
         if x.0 == 0 {
           |y| (0,y)
         } else {
           |y| (y,y)
         }
        };
f(5)
```

- => (5,5)

```
let f = {let x = (2,3);
          |y| (x.0,y)
        };
f(5)
```

- => ERROR! Use of x after free!

Annotating the program with regions (Tofte & Talpin, 1994)

```
letregion ρ4,ρ5 in
  let f =
    letregion ρ6 in
      let x = (2 at ρ2, 3 at ρ6) at ρ4;
          (|y| (x.0,y) at ρ1) at ρ5
  at ρ5;
  f(5 at ρ3)
```

Regions (lifetimes) in Rust

```
let f =                    //  9
  {let x = (2,3);          // 10
   |y| (x.0, y)            // 11
  };                       // 12
f(5)                       // 13
```

```
error[E0597]: `x` does not live long enough
  --> src/main.rs:11:23
   |
11 |         Box::new(|y| (x.0, y))
   |                  ---  ^ does not live long enough
   |                  |
   |                  capture occurs here
12 |     };
   |     - borrowed value only lives until here
13 |     println!("{:?}", f(5))
14 | }
   | - borrowed value needs to live until here
```

***** Compile-time error***

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." – rust-lang.org*
  - *"Rust is a systems language pursuing the trifecta: safe, concurrent, and fast" – This Week in Rust*

```
fn main() {
    let greetings = ["Hello", "Ciao", "こんにちは",
                     "안녕하세요", "Cześć", "Olá"];

    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
- Safe
  - Memory safety: no double frees, no NULL pointers, no dangling pointers, no scribblers
  - API safety: compile-time API checking, no “unexpected behaviour”

Other languages:

C/C++:
- Fast: yes
- Concurrent: yes
- Safe: no

Java/Scala:
- Fast: sort-of
- Concurrent: yes
- Safe: sort-of

Ownership in Rust

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

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

let v2 = v;

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

```
error: use of moved value: `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]);
```

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

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 s2 = sum_vec(v2);
        // Return the answer.
        s1 + s2
    }

    let v1 = vec![1, 2, 3];
    let v2 = vec![4, 5, 6];

    let answer = foo(&v1, &v2);
    println!("{}", answer);
}
```

Mutable references

```
let mut x = 5;
{  let mut x = 5;

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

```
error: cannot borrow `x` as immutable because it is also borrowed as mutable
    println!("{}", x);
                   ^
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;
                     ^
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 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

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;                // Introduce reference: `r`.
{
    let i = 1;        // Introduce scoped value: `i`.
    r = &i;           // Store reference of `i` in `r`.
}                     // `i` goes out of scope and is dropped.

println!("{}", r);  // `r` still refers to `i`.
```

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 main() {
    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

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

A dangerous API

```rust
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();
}
```

```rust
pub struct Letter {
    text: String,
}

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)*
Andrea Lattuada (ETH Zürich), RustFest 2017

Ownership types for API constraints

```
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 { } }
```

High-performance comms software

metaswitch

Metaswitch Networks

- Cloud-based communications software, powering more than 1,000 worldwide service provider networks.
- Commercial and open-source software solutions.
- Transforming communications networks – migration, virtualization, scale, reliability, manageability.

- 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*
- https://www.rust-lang.org/
- Wikipedia on Rust
- Rust Book (1st ed) bibliography - types
- The Rust Book (1st ed): chapter on ownership
- Metaswitch blog (e.g., 1, 2)

Contact me: Keith.Wansbrough@metaswitch.com

@kw217

Additional slides

Annotating the program with regions (Tofte & Talpin, 1994)

```
let f =

    let x = (2      , 3      )      ;
       (|y| (x.0,y)      )
    ;
f(5      )
```

Annotating the program with regions (Tofte & Talpin, 1994)

```
letregion ρ4,ρ5 in
  let f =
    letregion ρ6 in
      let x = (2 at ρ2, 3 at ρ6) at ρ4;
          (|y| (x.0,y) at ρ1) at ρ5
  at ρ5;
  f(5 at ρ3)
```

Region usage:

- ρ_4: x = (2,3) – *used during application*
- ρ_5: |y| (x.0, y) – *used during application*
- ρ_6: 3 – *not used: quickly destroyed*
- ρ_1: (2,5) – *result of computation*
- ρ_2: 2 – *part of result*
- ρ_3: 5 – *argument; part of result*

- Region allocation:
  - `letregion ρ in e`
  - `e at ρ`

$$f : Int \xrightarrow{get(\rho_4)} (Int, Int)$$

Tofte & Talpin's type rules (selected)

$$\frac{TE + \{x \mapsto \mu_1\} \vdash e \Rightarrow e' : \mu_2, \varphi \qquad \varphi \subseteq \varphi' \quad \tau = \mu_1 \xrightarrow{\epsilon.\varphi'} \mu_2 \quad \mathrm{frv}(e') \subseteq \mathrm{frv}(TE, \tau)}{TE \vdash \lambda x.e \Rightarrow \lambda x.e' \ \texttt{at} \ \rho : (\tau, \rho), \{\mathbf{put}(\rho)\}} \tag{23}$$

$$\frac{TE \vdash e_1 \Rightarrow e_1' : (\mu' \xrightarrow{\epsilon.\varphi} \mu, \rho), \varphi_1 \qquad TE \vdash e_2 \Rightarrow e_2' : \mu', \varphi_2}{TE \vdash e_1\ e_2 \Rightarrow e_1'\ e_2' : \mu, \varphi \cup \varphi_1 \cup \varphi_2 \cup \{\epsilon, \mathbf{get}(\rho)\}} \tag{24}$$

$$\frac{TE \vdash e \Rightarrow e' : \mu, \varphi \qquad \rho \notin \mathrm{frv}(TE, \mu)}{TE \vdash e \Rightarrow \texttt{letregion}\ \rho\ \texttt{in}\ e'\ \texttt{end} : \mu, \varphi \setminus \{\mathbf{put}(\rho), \mathbf{get}(\rho)\}} \tag{27}$$

Regions (lifetimes) in Rust

```
let f =                    //  9
  {let x = (2,3);          // 10
   |y| (x.0, y)            // 11
  };                       // 12
f(5)                       // 13
```

```
error[E0597]: `x` does not live long enough
  --> src/main.rs:11:23
   |
11 |         Box::new(|y| (x.0, y))
   |                  ---  ^ does not live long enough
   |                  |
   |                  capture occurs here
12 |     };
   |     - borrowed value only lives until here
13 |     println!("{:?}", f(5))
14 | }
   | - borrowed value needs to live until here
```

***** Compile-time error***

Passing back ownership?

```
fn foo(v: Vec<i32>) -> Vec<i32> {
```

```
fn foo(v1: Vec<i32>, v2: Vec<i32>) -> (Vec<i32>, Vec<i32>, i32) {
    // Do stuff with `v1` and `v2`.

    // Hand back ownership, and the result of our function.
    (v1, v2, 42)
}

let v1 = vec![1, 2, 3];
let v2 = vec![1, 2, 3];

let (v1, v2, answer) = foo(v1, v2);
```

Ugh! This is why Rust has *borrowing*.

- Intro
- Common vulnerabilities
- Linear types
- Region types
- Rust intro
- Ownership
- Lifetimes
- Summary
- API safety
- Metaswitch
- Conclusion