MesaLock Linux

Towards a memory-safe Linux distribution

Mingshen Sun
MesaLock Linux Maintainer | Baidu X-Lab, USA

Shanghai Jiao Tong University, 2018

whoami

- Senior Security Research in Baidu X-Lab, Baidu USA
- PhD, The Chinese University of Hong Kong
- System security, mobile security, IoT security, and car hacking
- MesaLock Linux, TaintART, Pass for iOS, etc.
- `mssun @ GitHub` | `https://mssun.me`

MesaLock Linux

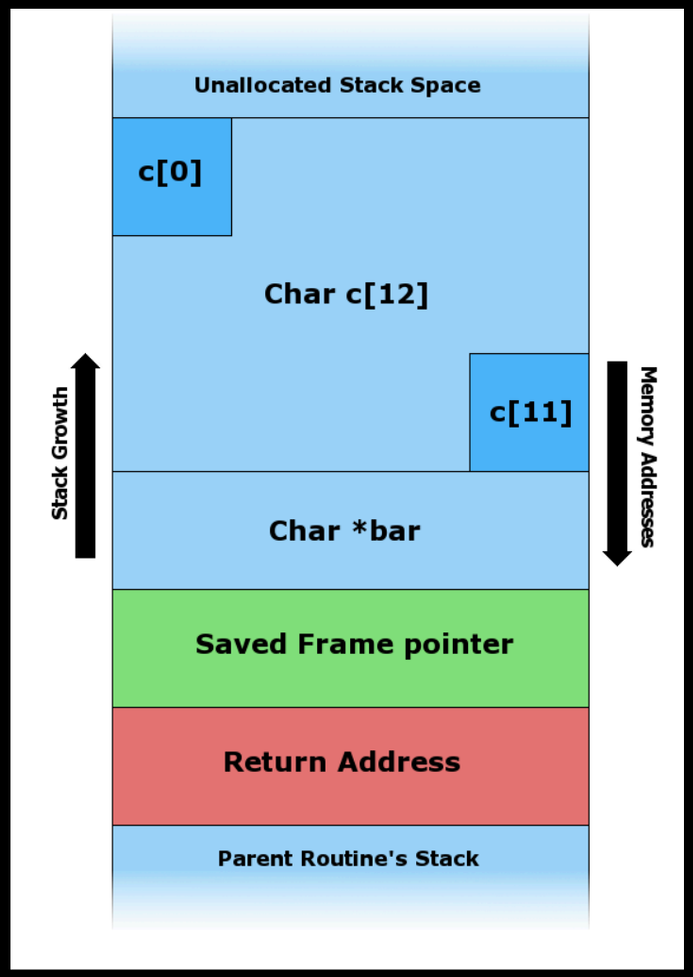
- Why
- What
- How

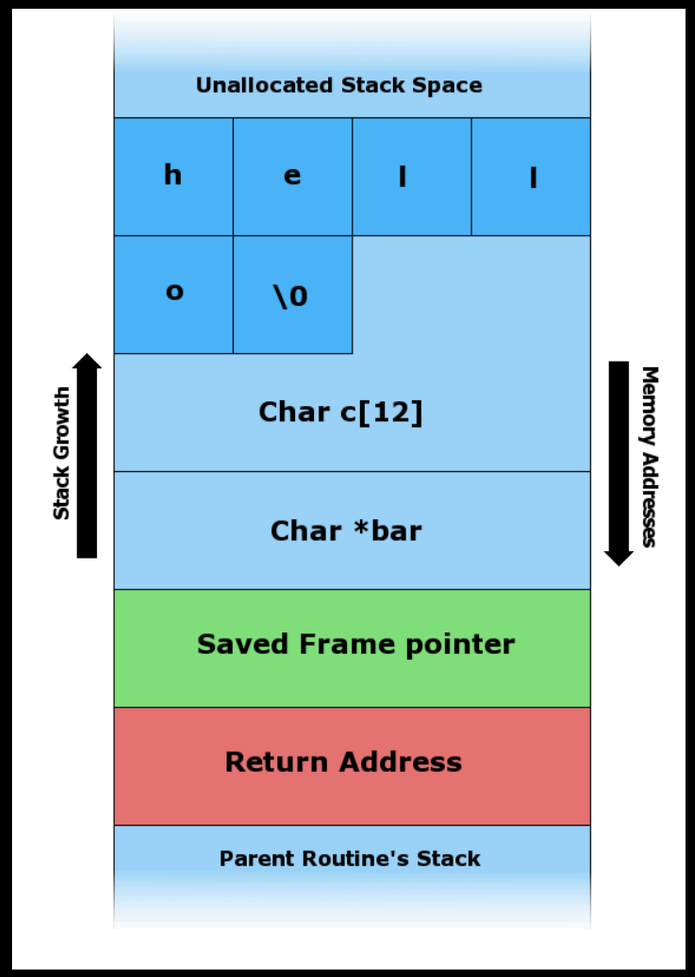
Why

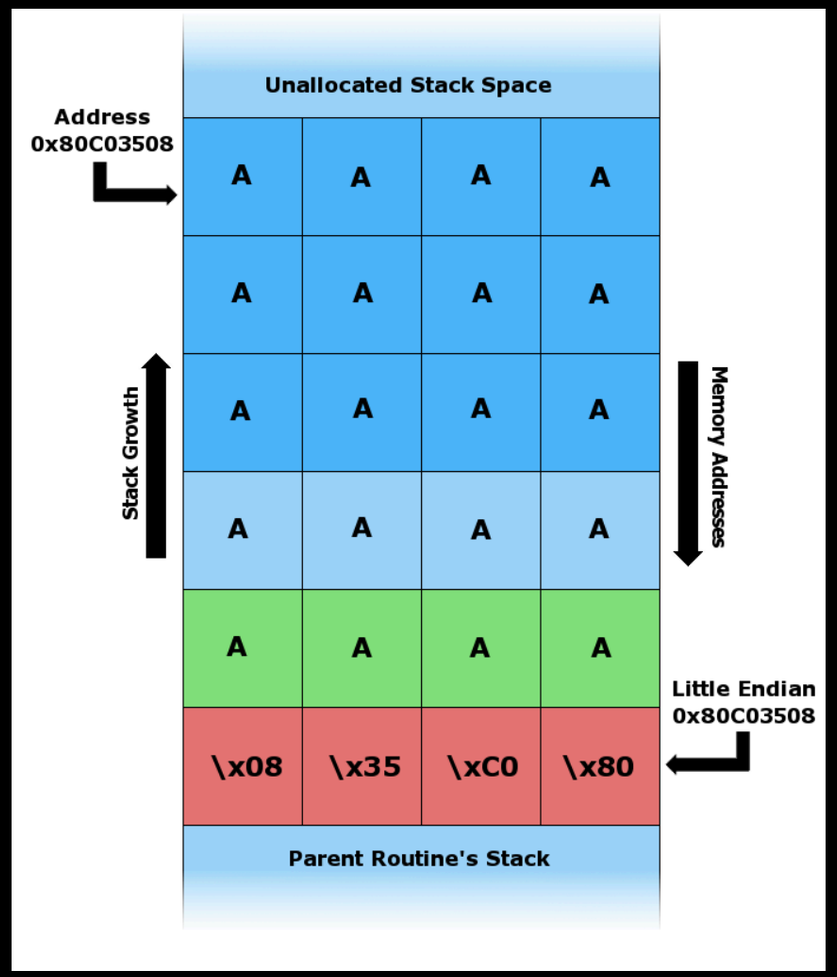
- **Memory corruption** occurs in a computer program when the contents of a memory location are **unintentionally modified**; this is termed violating memory safety.

- **Memory safety** is the state of being protected from various software bugs and security vulnerabilities when dealing with memory access, such as **buffer overflows and dangling pointers**.

Stack Buffer Overflow







- https://youtu.be/T03idxny9jE

Types of memory errors

- Access errors
  - Buffer overflow
  - Race condition
    - Use after free
- Uninitialized variables
- Memory leak
  - Double free

Memory-safety in user space

- CVE-2017-13089 **wget**: Stack-based buffer overflow in HTTP protocol handling

- A **stack-based buffer overflow** when processing chunked, encoded HTTP responses was found in wget. By tricking an unsuspecting user into connecting to a malicious HTTP server, an attacker could exploit this flaw to potentially **execute arbitrary code**.

- https://bugzilla.redhat.com/show_bug.cgi?id=1505444

- POC: https://github.com/r1b/CVE-2017-13089

What

- Linux distribution
- Memory-safe user space

Linux Distribution

- A Linux distribution (often abbreviated as distro) is an operating system made from a **software collection**, which is based upon the **Linux kernel** and, often, a **package management system**.

Linux Distros

- **Server**: CentOS, Federa, RedHat, Debian
- **Desktop**: Ubuntu
- **Mobile**: Android
- **Embedded**: OpenWRT, Yocto
- **Hard-core**: Arch Linux, Gentoo
- **Misc**: ChromeOS, Alpine Linux

Security and Safety?

- Gentoo Hardened: enables several risk-mitigating options in the toolchain, supports **PaX**, **grSecurity**, SELinux, **TPE** and more.
- Kernel hardening patches
- Safety? No.
- User space? GNU.

Introducing MesaLock Linux

- MesaLock Linux is a general purpose Linux distribution which aims to **provide a safe and secure user space environment**. To eliminate high-severe vulnerabilities caused by memory corruption, the whole user space applications are **rewritten in memory-safe programming languages like Rust and Go**.

Programming Language

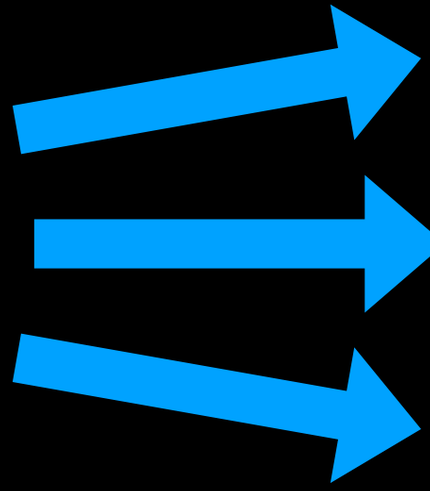
	Non-safe	Safe
Control	C/C++	Rust
Less control	Python	Go

Rust

- Rust is a **systems programming language** that runs blazingly **fast**, prevents segfaults, and guarantees **thread safety**.

How Does Rust Guarantee Memory Safety?

- Ownership
- Borrowing



- No need for a runtime (C/C++)
- Memory safety (GC)
- Data-race freedom

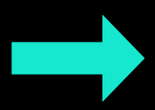
Ownership

Alice


```
fn main() {
    let alice = vec![1, 2, 3];;
    {
        let bob = alice;
        println!("bob: {}", bob[0]);
    }
    println!("alice: {}", alice[0]);
}
```

Ownership (T)

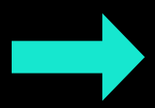
Alice **Bob**



```
fn main() {
    let alice = vec![1, 2, 3];;
    {
        let bob = alice;
        println!("bob: {}", bob[0]);
    }
    println!("alice: {}", alice[0]);
}
```

Ownership (T)

Alice **Bob**



```
fn main() {
    let alice = vec![1, 2, 3];;
    {
        let bob = alice;
        println!("bob: {}", bob[0]);
    }
    println!("alice: {}", alice[0]);
}
```

Ownership (T)

Alice


```
fn main() {
    let alice = vec![1, 2, 3];;
    {
        let bob = alice;
        println!("bob: {}", bob[0]);
    }
    println!("alice: {}", alice[0]);
}
```

Ownership (T)

Alice

```
error[E0382]: use of moved value: `alice`
 --> src/main.rs:7:27
  |
4 |         let bob = alice;
  |             --- value moved here
...
7 |     println!("alice: {}", alice[0]);
  |                           ^^^^^ value used here after move
  |
  = note: move occurs because `alice` has type
`std::vec::Vec<i32>`, which does not implement the `Copy` trait
```

```
fn main() {
    let alice = vec![1, 2, 3];;
    {
        let bob = alice;
        println!("bob: {}", bob[0]);
    }
    println!("alice: {}", alice[0]);
}
```

Ownership (T)

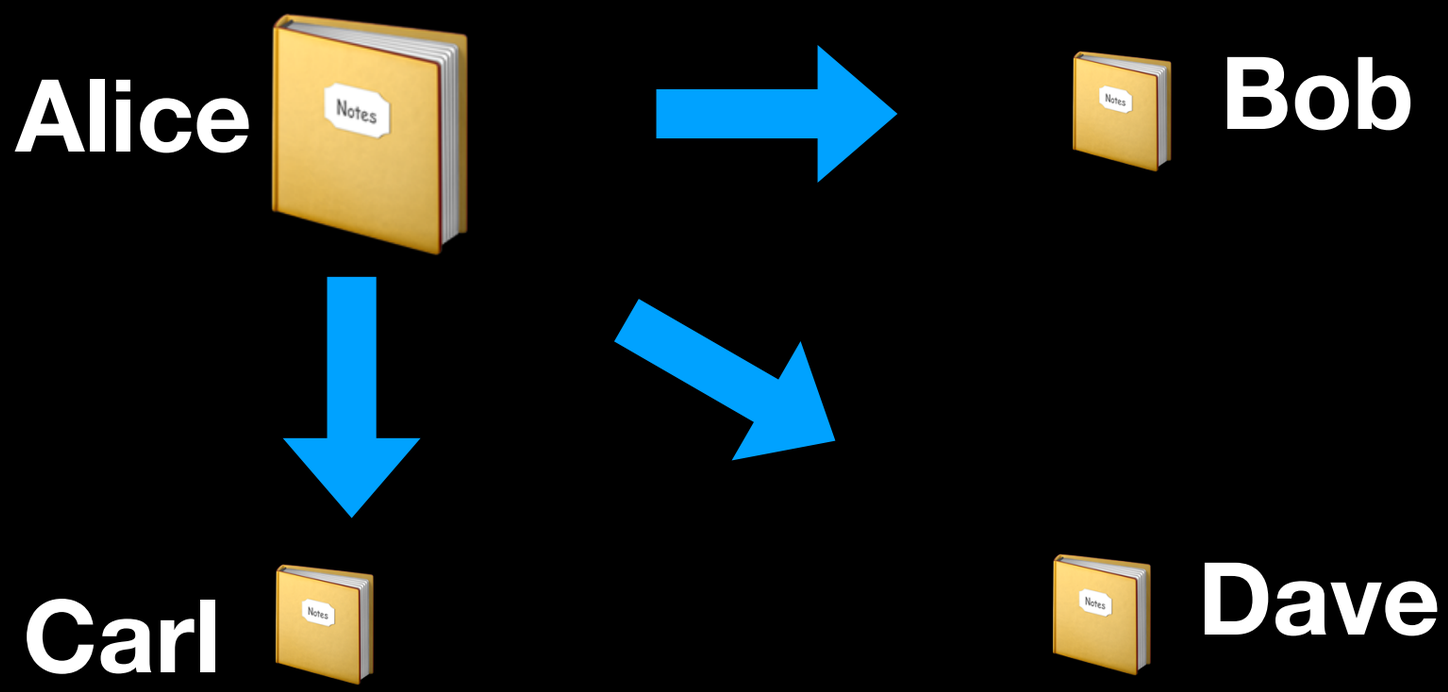
Alice

```
error[E0382]: use of moved value: `alice`
 --> src/main.rs:7:27
  |
4 |         let bob = alice;
  |             --- value moved here
...
7 |     println!("alice: {}", alice[0]);
  |                           ^^^^^ value used here after move
  |
  = note: move occurs because `alice` has type
`std::vec::Vec<i32>`, which does not implement the `Copy` trait
```

```
fn main() {
    let mut alice = vec![1, 2, 3];;
    {
        let mut bob = alice;
        println!("bob: {}", bob[0]);
    }
    println!("alice: {}", alice[0]);
}
```

Shared Borrow (&T)

Aliasing + ~~Mutation~~



Mutable Borrow (&mut T)

Alice


```
fn main() {
    let mut alice = 1;
    {
        let bob = &mut alice;
        *bob = 2;
        println!("bob: {}", bob);
    }
    println!("alice: {}", alice);
}
```

Mutable Borrow (&mut T)

Alice **Bob**


```
fn main() {
    let mut alice = 1;
    {
        let bob = &mut alice;
        *bob = 2;
        println!("bob: {}", bob);
    }
    println!("alice: {}", alice);
}
```

Mutable Borrow (&mut T)

Alice


```
fn main() {
    let mut alice = 1;
    {
        let bob = &mut alice;
        *bob = 2;
        println!("bob: {}", bob);
    }
    println!("alice: {}", alice);
}
```

Mutable Borrow (&mut T)

Alice


```
fn main() {
    let mut alice = 1;
    {
        let bob = &mut alice;
        *bob = 2;
        println!("bob: {}", bob);
    }
    println!("alice: {}", alice);
}
```

Mutable Borrow (&mut T)

~~Aliasing~~ + Mutation

Alice

The lifetime of a borrowed reference should end before the lifetime of the owner object does.

Rust's Ownership & Borrowing

~~Aliasing~~ + ***~~Mutation~~***

- Compiler enforced:
  - Every resource has a unique **owner**
  - Others can **borrow** the resource from its owner (e.g., create an **alias**) with restrictions
  - Owner **cannot** free or mutate its resource while it is borrowed

Use-After Free in C/Rust

C/C++

```
void func() {
    int *used_after_free = malloc(sizeof(int));

    free(used_after_free);

    printf("%d", *used_after_free);
}
```

Rust

```
fn main() {
    let name = String::from("Hello World");
    let mut name_ref = &name;
    {
        let new_name = String::from("Goodbye");
        name_ref = &new_name;
    }
    println!("name is {}", &name_ref);
}
```

Use-After Free in Rust

```
error[E0597]: `new_name` does not live long enough
 --> main.rs:7:5
  |
6 |         name_ref = &new_name;
  |                    --------- borrow occurs here
7 |     }
  |     ^ `new_name` dropped here while still borrowed
8 |     println!("name is {}", &name_ref);
9 | }
  | - borrowed value needs to live until here

error: aborting due to previous error
```

Formal Verification

- RustBelt: Securing the Foundations of the Rust Programming Language (POPL 2018)

- *In this paper, we give the first **formal (and machine-checked) safety proof** for a language representing a realistic subset of Rust.*

- https://people.mpi-sws.org/~dreyer/papers/rustbelt/paper.pdf

Rust's Performance vs C

The Computer Language Benchmarks Game

Rust programs versus C gcc

all other Rust programs & measurements

by benchmark task performance

k-nucleotide

source	secs	mem	gz	cpu	cpu load
Rust	**4.86**	137,656	1748	15.42	57% 100% 75% 88%
C gcc	6.67	130,160	1506	19.33	76% 50% 92% 75%

reverse-complement

source	secs	mem	gz	cpu	cpu load
Rust	**0.37**	250,708	1376	0.66	100% 28% 28% 31%
C gcc	0.48	200,492	820	0.66	4% 19% 25% 98%

spectral-norm

source	secs	mem	gz	cpu	cpu load
Rust	**1.98**	2,580	817	7.90	100% 99% 100% 99%
C gcc	2.00	1,300	569	7.89	99% 99% 100% 99%

pidigits

source	secs	mem	gz	cpu	cpu load
Rust	**1.74**	4,532	1366	1.74	100% 1% 0% 2%
C gcc	1.74	2,716	452	1.74	100% 1% 1% 2%

Rust's Performance vs Go

The Computer Language Benchmarks Game

Rust programs versus Go

all other Rust programs & measurements

by benchmark task performance

regex-redux

source	secs	mem	gz	cpu	cpu load
Rust	**2.99**	192,244	804	4.36	31% 16% 85% 16%
Go	28.49	318,144	802	59.96	74% 48% 41% 48%

binary-trees

source	secs	mem	gz	cpu	cpu load
Rust	**3.88**	173,136	721	14.24	90% 91% 100% 90%
Go	34.42	268,188	654	130.26	96% 94% 95% 94%

k-nucleotide

source	secs	mem	gz	cpu	cpu load
Rust	**4.86**	137,656	1748	15.42	57% 100% 75% 88%
Go	14.98	147,704	1722	55.60	97% 88% 92% 95%

mandelbrot

source	secs	mem	gz	cpu	cpu load
Rust	**2.02**	33,572	1332	8.00	99% 100% 99% 99%
Go	5.48	30,704	905	21.74	100% 100% 99% 100%

Rust's Performance

- Firefox Quantum includes Stylo, a pure-Rust CSS engine that makes full use of Rust's "**Fearless Concurrency**" to speed up page styling. It's the first major component of Servo to be integrated with Firefox, and is a major milestone for Servo, Firefox, and Rust. *It replaces approximately 160,000 lines of C++ with 85,000 lines of Rust.*

- Parallelism leads to a lot of performance improvements, including a **30% page load speedup** for Amazon's homepage.

Go?

- Type safe
- Fast GC
- Good at parallelization

MesaLock Linux

- Linux kernel
  - Compatible
  - Stable
- Memory-safe user space
  - Safe
  - Secure

Rules-of-thumb for Hybrid Memory-safe Architecture

- Unsafe components should be appropriately **isolated** and **modularized**, and the size should be small (or minimized).

- Unsafe components **should not weaken** the safe, especially, public APIs and data structures.

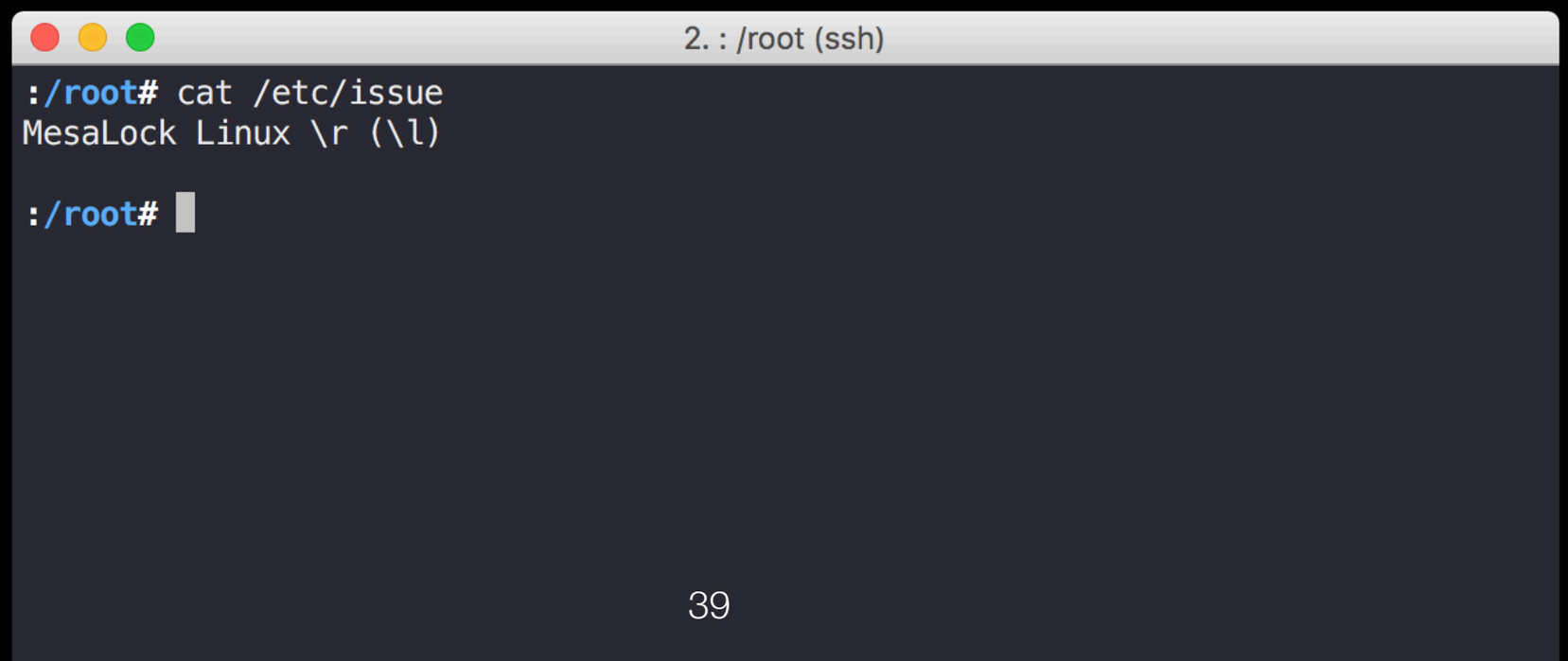
- Unsafe components should be **clearly identified** and **easily upgraded**.

MesaLock Linux

- Live ISO
- Docker image
- rootfs
- x86_64, arm in the near future

Quick Start

```
$ docker run -it mesalocklinux/mesalock-linux
```



MesaLock Linux From Scratch

- Bootloader
- Linux kernel
- init
- getty
- login
- iproute2
- coreutils
- syslinux
- Linux 4.9.58
- minit (Rust)
- mgetty (Rust)
- giproute2 (Go)
- uutils-coreutils (Rust)

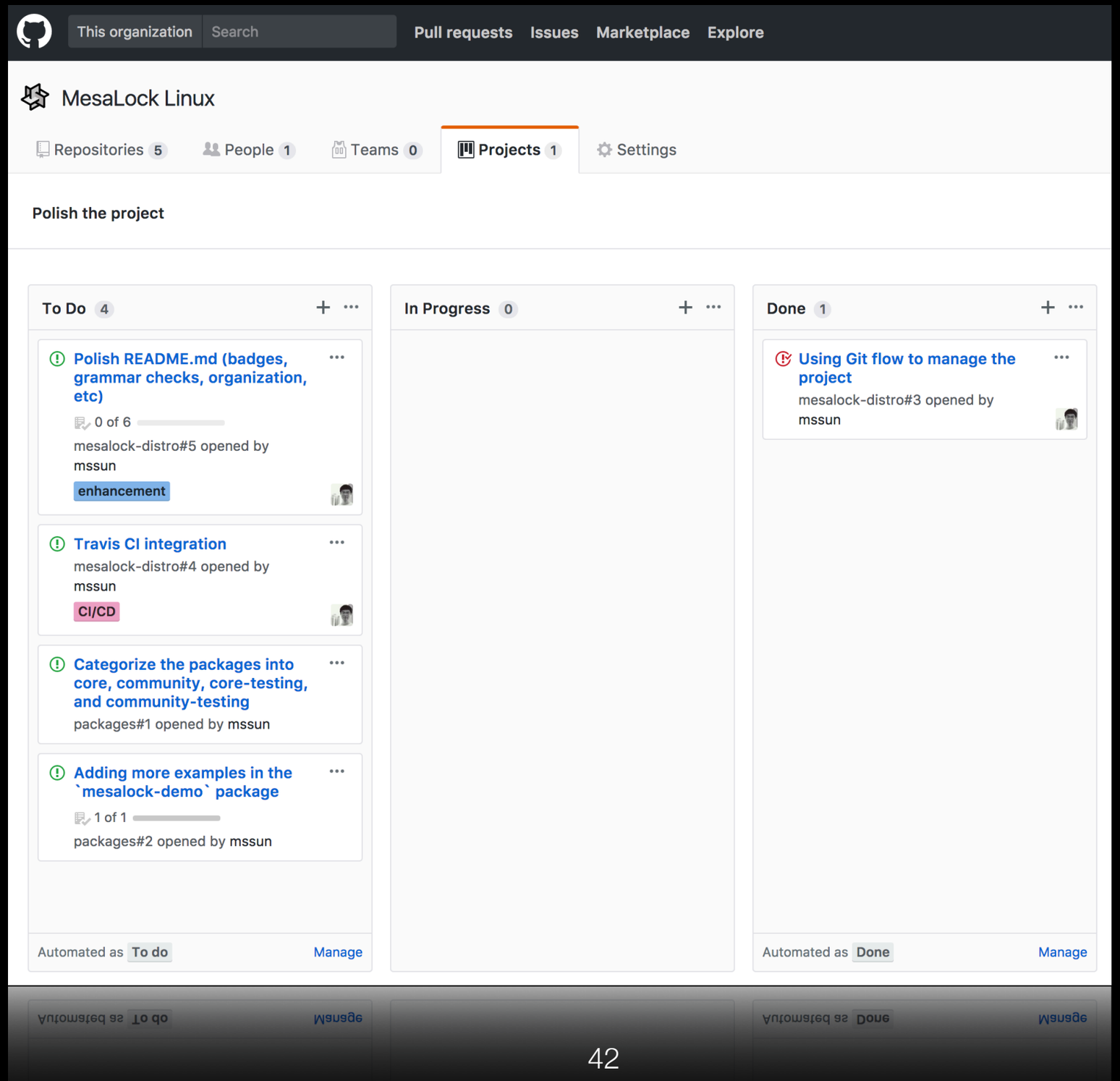
The MesaLock Linux Project

- Open source
  - BSD License (friendly)
  - Development (commit history)
  - Issue (GitHub issue tracking)
  - Discussion (IRC)
  - Roadmap (open)
  - etc

http://mesalock-linux.org

#mesalock-linux
#mesalock-linux-cn
#mesalock-linux-devel
@ Freenode

The MesaLock Linux Project



The MesaLock Linux Project

- Two parts:
  - MesaLock Linux: building scripts, etc
  - Core packages: `minit`, `mgetty`, `giproute2`, etc

The MesaLock Linux Project

- More specific:
  - `mesalock-linux/mesalock-distro`
  - `mesalock-linux/packages`
  - `mesalock-linux/giproute2`
  - `mesalock-linux/minit`
  - `mesalock-linux/mgetty`
  - `mesalock-linux/miproute2`

The MesaLock Linux Packages

- `brotli`: compression tool written in Rust
- `busybox`: busybox tool set for testing only
- `exa`: replacement for ls written in Rust
- `fd-find`: simple, fast and user-friendly alternative to find
- `filesystem`: base filesystem layout
- `gcc-libs`: GCC library, only libgcc_s.so is used
- `giproute2`: ip tool written in Go
- `glibc`: glibc library
- `init`: init script
- `ion-shell`: shell written in Rust
- `linux`: Linux kernel

The MesaLock Linux Packages

- `mesalock-demo`: some demo projects
- `mgetty`: getty written in Rust
- `micro`: modern and intuitive terminal-based text editor in written Go
- `minit`: init written in Rust
- `ripgrep`: ripgrep combines the usability of The Silver Searcher with the raw speed of grep, written in Rust
- `syslinux`: bootloader
- `tokei`: count your code, quickly, in Rust
- `tzdata`: timezone data
- `uutils-coreutils`: cross-platform Rust rewrite of the GNU coreutils
- `uutils-findutils`: rust implementation of findutils
- `xi-core`: a modern editor with a backend written in Rust
- `xi-tui:` a tui frontend for Xi

New Package?

- A package consist of a `BUILD` script and related files and patches.
- The build tool (`mkpkg`) will call following function in order:

1. `prepare()`: downloading source code and prepare configration stuff
2. `build()`: buiding sources
3. `package()`: zip the output as a package

New Package?

2. vim ripgrep/BUILD (ssh)

```
pkgname=ripgrep
pkgver=0.7.1
pkgdesc="ripgrep combines the usability of The Silver Searcher with the raw
speed of grep."
url="https://github.com/BurntSushi/ripgrep"
license=(MIT Unlicense)

prepare() {
  git clone -b 0.7.1 https://github.com/BurntSushi/ripgrep.git "$srcdir"
}

build() {
  cd "$srcdir" && ./compile
}

package() {
  install -D "$srcdir"/target/release/rg -t "$pkgdir"/bin/
}
~
~
~
~
~
~
                                                    1,1           All
```

Roadmap

- 0.1: public release
- 0.2: polish source code organization, improved development process
- 0.3: improving core packages
- 0.4: including more utilities
- 0.5: support multi-platforms
- ...
- 1.0: in production

Contributing

- You can get involved in various forms:
  - Try to **use** MesaLock Linux, report issue, enhancement suggestions, etc
  - **Contribute to MesaLock Linux**: optimize development process, improve documents, closing issues, etc
  - **Contribute to core packages of MesaLock Linux**: improving minit, mgetty, giproute2, etc
  - **Writing applications** using memory safe programming languages like Rust/Go, and joining the the MesaLock Linux packages
  - **Auditing source code** of the MesaLock Linux projects and related packages
- You are welcome to send **pull requests** and report **issues** on GitHub.

Feedbacks

微博搜索 综合 找人 文章 视频 图片

MesaLock Linux

热门

keithcool

百度安全实验室MesaLock Linux开源，这是一个通用 Linux 发行版本，其目标是用 Rust、Go 等内存安全语言重写全部用户空间应用（user space applications），以在用户空间中逐步消除高危的内存安全漏洞。这将极大的降低整个系统的攻击面，并且使得剩余的攻击面可审计、可收敛。

MesaLock L...

MesaLock Linux 开源：一个用户空间内...

MesaLock Linux 是一个通用 Linux 发行版本，其目标是用 Rust、Go 等内存安全语言重写用户空间应用（user space applications），以在用户

12月2日18:53 来自 微博 weibo.com

收藏 | 转发 147 | 评论 22 | 43

按热度 | 按时间

老爷我瞌睡：这比那些“国产操作系统”有意义多了啊！

12月3日 15:39　　回复 | 9

keithcool 赞了这条评论

boywhp：会不会性能很差？用这些脚本语言？

12月3日 17:11　　回复 | 1

keithcool：Rust/Go都是编译语言...

12月3日 17:15　　回复 | 赞

胡争辉：重写整个GNU?

12月7日 18:02　　回复 | 赞

keithcool：是整合加补足，逐步形成一个完整的GNU的内存安全替代方案

12月8日 09:03　　回复 | 赞

胡争辉：回复@keithcool:谢谢

12月8日 23:08　　回复 | 赞

paiSing刘海舰：百度终于办了件有意义的事儿！

12月3日 20:21　　回复 | 赞

keithcool：现在微博配图的size不能自动适配，发个logo也被打败了

12月3日 10:53　　回复 | 赞

安全_云舒：干得漂亮。就看是否一直检查下去了。

12月3日 20:16　　回复 | 赞

Lucene田春峰

我的第一份 mesalock-linux iso 编译出来了。

mesalock 是一个内存安全的 linux 。

网页链接

非常棒!

12月4日15:30 来自 微博 weibo.com

收藏 | 转发 3 | 评论 7 | 3

This repository Search Pull requests Issues Marketplace Explore

mesalock-linux / mesalock-distro

Unwatch 9 Unstar 73 Fork 6

Code Issues 2 Pull requests 0 Projects 0 Wiki Insights Settings

Stargazers

All 73 You know 2

wangxianfeng
Joined on Dec 24, 2012
Follow

Kai Yao
Joined on Nov 8, 2015
Follow

pzasdq
Joined on Jan 3, 2016
Follow

netcon
Joined on Mar 8, 2016
Follow

Adam Yi
Joined on Aug 26, 2013
Follow

xiaobo55x
Joined on Oct 18, 2013
Follow

whjl2015
Joined on Aug 18, 2013
Follow

xiaolong321
Joined on Apr 7, 2016
Follow

hammerorz
Joined on Apr 2, 2017
Follow

stepbystep2
Joined on Oct 20, 2016
Follow

darkdown
Joined on Aug 9, 2014
Follow

mangoyuan
Joined on Mar 18, 2016
Follow

Thank you!

Future Work

- TrustZone OS in Rust
- Linux kernel driver in Rust

Memory-safety in Linux kernel

Vulnerability Trends Over Time

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
1999	19	7		3						1		2			
2000	5	3										1			
2001	23	7								4		3			
2002	15	3		1						1	1				
2003	19	8		2						1	3	4			
2004	51	20	5	12							5	12			
2005	133	90	19	19	1					6	5	7			
2006	90	61	5	7	7			2		5	3	3			
2007	63	41	2	8						3	8	7			
2008	71	44	3	17	4					4	6	11			
2009	105	66	2	22	7					8	11	22			5
2010	124	67	3	16	7					8	30	14			5
2011	83	62	1	21	10					1	21	9			1
2012	115	83	4	25	10					6	19	11			
2013	189	101	6	41	13					11	57	26			7
2014	133	89	8	21	10					11	30	20			10
2015	86	55	6	15	4					11	10	17			
2016	217	153	5	38	18					12	35	52			1
2017	421	129	168	40	18			1		14	85	33			
Total	1962	1089	237	308	109			3		107	329	254			29
% Of All		55.5	12.1	15.7	5.6	0.0	0.0	0.2	0.0	5.5	16.8	12.9	0.0	0.0	

Warning : Vulnerabilities with publish dates before 1999 are not included in this table and chart. (Because there are not many of them and they make the page look bad; and they may not be actually published in those years.)

Memory-safe Linux Device Driver using Rust

- https://github.com/tsgates/rust.ko
- a minimal Linux kernel module written in rust
- FFI, unsafe