POLYVERSE

Secure Your Operating Systems in Plain Sight

The Problem Defined

Today, security is often considered a developer burden. Why? Security is hard: it requires developers and testers to take time and energy away from working on core customer solutions. Security is critically important, yet when delivery deadlines are tight, security training and reviews are high on the list of items to be rescheduled.

Can we approach the problem of built-in security in a better way?

We think the answer is an emphatic "Yes."

Polymorphic Linux® versions of Red Hat, Centos, Ubuntu, Fedora, and Alpine make attacks such as buffer overflows, memory-exploits, and even CPU timing attacks impractical. How? Polyverse hides your operating system in plain sight. Polyverse compiles and scrambles the structure of the operating system (OS) binaries loaded into memory; the result is that each OS you use is unique yet the same. From a developer and user perspective, nothing about the way you use it has changed.

The difference is that attackers cannot apply common memory and register assumptions to each OS attack. Polyverse most notably does not change the source code of the OS in any way, only the compiled binary OS image. This approach drastically reduces the attack surface for the attacker, yet requires no changes to developer or user behavior.

Security today can be complicated and time-consuming. This paper demonstrates how to integrate Polymorphic Linux into a GitHub project, giving your projects powerful yet simple security protection. In this case, we use docker-android, which is maintained by Budi Utomo to demonstrate Polyverse integration. Why? Polyverse is committed to supporting the open-source community that supplies the tools that enable the Internet.

Budi Utomo kindly agreed to work with us in integrating Polymorphic Linux into his project to increase the project security, and in so doing highlight the ease of project updates and ease of integration with a set of Docker containers.

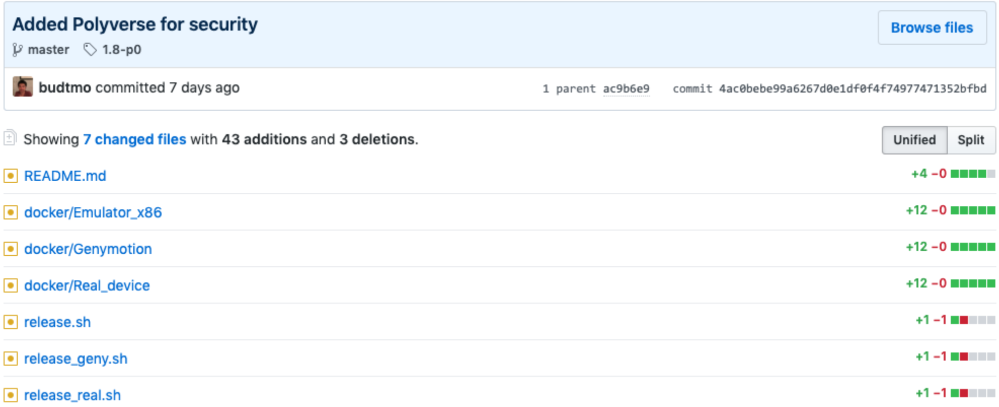
The Solution Implementation

The Polymorphic Linux version of Red Hat, Centos, Ubuntu, Fedora and Alpine can be installed using one command, making it simple to add into Scripts, Makefiles or, in this case, Dockerfiles.

```
curl https://sh.polyverse.io | sh -s install
czcw7pjshny8lzzog8bgiizfr demo@polyverse.io
```

To finish, you reinstall your installed packages. The command for this will vary depending on your Linux distribution, and our script will give you a simple command to perform the required package re-installation.[1] When reinstalling is complete you are protected by Polyverse![2]

Code Changes



Added Polyverse for security

master 1.8-p0

budtmo committed 7 days ago 1 parent ac9b6e9 commit 4ac0bebe99a6267d0e1df0f4f74977471352bfbd

Showing **7 changed files** with **43 additions** and **3 deletions**.

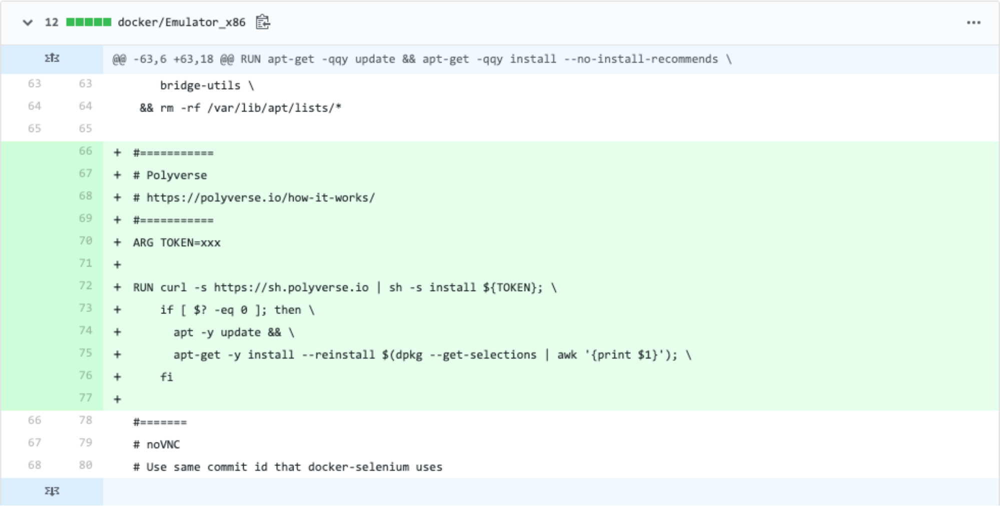
README.md	+4 −0
docker/Emulator_x86	+12 −0
docker/Genymotion	+12 −0
docker/Real_device	+12 −0
release.sh	+1 −1
release_geny.sh	+1 −1
release_real.sh	+1 −1

For the purposes of a Docker container there are several places you could insert our installation script: in docker-compose.yml, in the dockerfile in the entry point, the CMD nodes, or in a RUN command in the dockerfile.

In this case, we went with a RUN command in the dockerfile.
By inserting the install command here, Polyverse's scrambled binaries are downloaded when the docker container image is built. Replacing all the binaries on an image can take time, so instead of replacing them when the container is started, it's more efficient to do it once at build time.

[1] You can also find these commands here: https://polyverse.io/polymorphic-linux-installation-guide/

[2] For this case study, we're using the publicly available demo auth key, but we have an open-source specific auth key for supporting projects. Contact sales@polyverse.io for more information about how we can support your project.



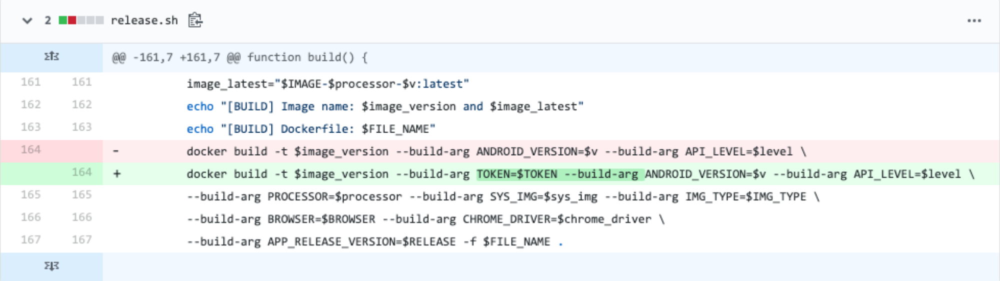
```
@@ -63,6 +63,18 @@ RUN apt-get -qqy update && apt-get -qqy install --no-install-recommends \
      bridge-utils \
   && rm -rf /var/lib/apt/lists/*

+ #===========
+ # Polyverse
+ # https://polyverse.io/how-it-works/
+ #===========
+ ARG TOKEN=xxx
+
+ RUN curl -s https://sh.polyverse.io | sh -s install ${TOKEN}; \
+     if [ $? -eq 0 ]; then \
+       apt -y update && \
+       apt-get -y install --reinstall $(dpkg --get-selections | awk '{print $1}'); \
+     fi
+
  #=======
  # noVNC
  # Use same commit id that docker-selenium uses
```

We wrap the command that replaces the binaries in a conditional to ensure that if something goes wrong during the install process, the entire command doesn't return a non-zero result. This ensures that Polyverse isn't a hard dependency and operations remain unaffected. In the Github implementation, the Polyverse authentication key is replaced by ${TOKEN} and is resolved upon execution.

The command used was

```
RUN curl -s https://sh.polyverse.io | sh -s install
czcw7pjshny8lzzog8bgiizfr; \
    if [ $? -eq 0 ]; then \
      apt -y update && \
      apt-get -y install --reinstall $(dpkg \
      --get-selections | awk ‘{print $1}’);
    fi
```

For the purposes of docker-android, we rebuild the local docker-android images in order to see the changes made in the dockerfile. To do this, we run the release.sh script in the root of the project and selected the build task for all Android versions.



```
@@ -161,7 +161,7 @@ function build() {
          image_latest="$IMAGE-$processor-$v:latest"
          echo "[BUILD] Image name: $image_version and $image_latest"
          echo "[BUILD] Dockerfile: $FILE_NAME"
-         docker build -t $image_version --build-arg ANDROID_VERSION=$v --build-arg API_LEVEL=$level \
+         docker build -t $image_version --build-arg TOKEN=$TOKEN --build-arg ANDROID_VERSION=$v --build-arg API_LEVEL=$level \
          --build-arg PROCESSOR=$processor --build-arg SYS_IMG=$sys_img --build-arg IMG_TYPE=$IMG_TYPE \
          --build-arg BROWSER=$BROWSER --build-arg CHROME_DRIVER=$chrome_driver \
          --build-arg APP_RELEASE_VERSION=$RELEASE -f $FILE_NAME .
```

Testing

Docker includes a health check for containers so you can monitor crashes and freezes in order to direct traffic to a different container if necessary. To test, we confirm that the health checks still work after we have added Polymorphic Linux to the Docker images. All health checks continue to work as expected and the test passed.

It is also necessary to test the connection to the emulators running in each Docker container using noVNC. This works exactly as before. Finally, we make sure that the binaries installed on the system are actually downloaded from Polyverse's repositories. To do this we start a bash prompt on a running container using this command:

```
docker exec -it <container name> /bin/bash
```

When run, dpkg shows where each binary is downloaded from. The Docker-android's Appium container structure is based on Ubuntu 18.04, so we can use the command below to list the source of each package:

```
dpkg --get-selections | awk '{print $1}' | xargs
apt-cache madison | grep -i polyverse
```

Polyverse also provides a script that has the same function but is independent of the distro you are using. We use the `list-installed` command to list the binaries that come from Polyverse's repositories:

```
curl https://sh.polyverse.io | sh -s list-installed |
grep polyverse
```

By running the `list-installed` command we confirm that the originally installed packages are replaced with Polyverse's scrambled binaries.

Conclusion and Benefits

Today, security is often regarded as a burden and ignored, as it takes away time and energy spent on creating new technology. By successfully integrating Polyverse into an open-source project with minimum fuss:

- We demonstrate that security does not have to be complicated and time-consuming.
- We integrate Polymorphic Linux into the project without changing the way docker-android is used.
- We demonstrate that Polymorphic Linux is easy to use and accessible to anyone, and that OS Logging, GDB debugging and all other functionalities behave the same way.

The binaries stay semantically identical, meaning defenders can install a Polymorphic Linux version of Red Hat, Centos, Ubuntu and essentially forget about it. Polyverse shifts the burden dramatically from the defender to the attacker. Polyverse increases work for the attacker, not the defender. In fact it makes the attacker's job all but impossible.

Further, there is no detectable performance impact, nor do you have to change your developer's programming habits to get value from Polymorphic Linux. Our goal is to give our customers more time to be responsive to business issues. By greatly reducing your security exposure we give you that time.

We believe in the open-source community and as we use open-source technology to build our product giving back is important to us. The community is more powerful than any single company, and only through the collaboration of many individuals and entities will we solve cybersecurity for good.

Have an open-source project? Contact shaina@polyverse.io to get Polymorphic Linux builds free of charge.