Productizing Linux Applications

Peter Ashford
Ashford Computer Consulting Service
9/08/2015.

What We'll Cover

The first part of this presentation describes a process that will assist you in delivering updates to a Linux Application, delivered as either a software appliance or a hardware appliance.

This is basically a continuous integration process for a (mostly) stable application on a rapidly changing Operating System.

It will also work for a DevOps environment.

What We'll Cover

The second part of this presentation describes methods that can be used to protect your Intellectual Property.

These same methods will help to protect your customer's data.

Part 1

Delivering updates to field systems

Process Goals

The goals of this process are:

- Perform final integration of an application into an Operating System on a frequent basis.
- Test and Verify proper functionality prior to releasing patches to the customer.
- Functions as a last chance to catch critical bugs and compatibility issues before distribution.

Overview

There are four general steps to this process.

1. Update local repository from OS provider.
2. Test current Application against updated OS.
3. Migrate necessary OS packages to distribution repository.
4. Create ISO for reinstallation.

NOTE: Optional special case for kernel updates on hardware appliances.

Update Local Repository

- The local repository should be updated at least once a day from the OS provider.
- If the OS provider allows a check for “Security” updates, then checks should be made more often (every few hours).
- When the OS provider releases an update, the files changed should be checked against the list of files needed for a release of the appliance. If any of the release files have been updated, a snapshot should be made for testing use.

Test the Application

The application needs to be tested against the updated OS.

At least two test environments need to be used.

1. Initial install using new OS.
2. Update good running system.

Critical configuration data should be preserved.

Testing Notes

- Automated Unit Testing should be performed.
- Automated Integration Testing should be performed.
- When an Application bug is fixed, additional tests will be needed.
- Even with the automated testing, it is still appropriate to perform full manual testing from time to time.

Migrate Packages to Distribution

1. Copy the changed packages into the distribution tree.
2. Take a snapshot of distribution repository.
3. Change the link for distribution.

Exceptions

When a test fails, the system will have to stop performing updates, and inform people, so that the problem can be resolved.

Kernel Updates

- Kernel updates are rarely actually needed for appliances.
- Most kernel updates are for new or updated drivers.
- They need to be sent out anyway, to keep in sync with packages that support and interact with the kernel.

Kernel Updates - Hardware

A hardware appliance can have the kernel minimized to only support those devices that are actually part of the appliance. This can make software theft more difficult.

It is also possible to build a kernel that contains only the appropriate drivers, and won't accept modules.

Persistent Configuration

In order to allow a reinstallation of the software to be used as a fallback for resolving problems, it is appropriate to keep the software configuration in a location that can be preserved during a system installation. The simplest such location is on a dedicated partition of the system disk.

This requires an installation option that can preserve the configuration, and an integration test to verify that the configuration is, in fact, kept and used.

Why?

Why do all this extra work?

- Delivering security patches to the customer is becoming a high priority in the industry.
- It is best to patch and test frequently, to minimize chances of intermediate patches causing delays at critical times.
- Reduces accumulation of Technical Debt.

Simplify

This is still a lot of work. How can we make it simpler?

- Only release to customers when the application changes or for OS Security updates.

Simplify (cont.)

- Reduce the number of Linux packages that are part of your appliance.
  - Reduces the installation & update time.
  - May reduce number of necessary tests.
  - May reduce attack surface of appliance.
  - Reduces storage and bandwidth needed for distribution.
  - May reduce appliance boot time.

Part 2

Protecting your Intellectual Property

Software

The delivered software should be compiled whenever possible.

While compiled software can be decompiled, it is more complex than just reading a script.

When software can't be compiled, it should be obfuscated.

Licenses

- License keys are usually cryptographically generated.
- License keys can be time-limited or perpetual.
- License keys can be delivered by hardcopy, phone or email.
- Many Linux-based software packages are using the FlexLM license manager to manage their license keys.
- If a Demo license is used to create the data, it may be appropriate to not allow it to read a previous configuration on an initial install.

Client Authentication

The distribution server can be set up to authenticate clients by the use of an SSL certificate. The certificate would contain the SSN (System Serial Number).

Downloads would be logged by requesting IP address, file and SSN.

Logs could be processed to look for duplicate downloads of packages by same SSN.

Use Hardware Authentication

It's possible to use the hardware as part of the authentication process.

- The MAC address of the primary NIC is a common unique value to use to identify a system.
- It is also possible to store an SSL certificate on either the TPM built into most motherboards or a USB dongle.

Whitelist Applications

It's possible to limit the applications that are run on the system by the use of application whitelisting.

- Use AppArmor

Copyright Violations

Inevitably, Copyright violations will occur. It is up to the Copyright holder to determine the appropriate response to these violations.

- Keep application from functioning – OK
- Uninstall application – OK
- Harm OS instance – NOT OK
- Harm data – NOT OK

Security Concerns

Security problems with your software are likely to make it easier for attackers to steal your IP.

- Don’t put a back door into your appliance.
- Don’t hard-code passwords into your application.
- Implement a method to limit incoming management connections.

Security Concerns (cont.)

Simple security problems with your software are likely to make it easier for attackers to steal your IP.

- Always validate user supplied data.
- Avoid buffer overflows when copying data.
- Never store passwords in plaintext. Store them as hashes.

Security Concerns (cont.)

Complex security problems with your software may make it easier for attackers to steal your IP.

- Instead of system(3), use fork(2)/exec(2).
- Run as little as possible as “root”.
- Work within an SELinux Context whenever possible.

Feedback

My goal is to improve this presentation. To that end, I would appreciate feedback to:

- ashford@ACCS.com
- www.linkedin.com/in/peterashford