Software Defined Networking Security

Outline

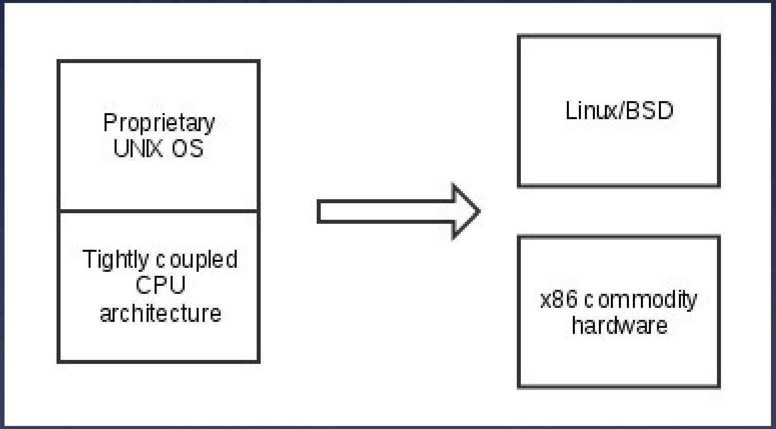
Introduction

- Security nerd, recovering climatologist
- Managed Red Hat's Java middleware security team
- Now manager of product security for Console, and founder of the ODL and ONOS security teams
- Open source SDN is hot, with development being driven by a wide range of commercial and non-profit entities
- 2015 is emerging as the year when SDN starts to move from the lab to widespread deployment for production networks (Google, Pacnet, etc.)
- Is it secure?

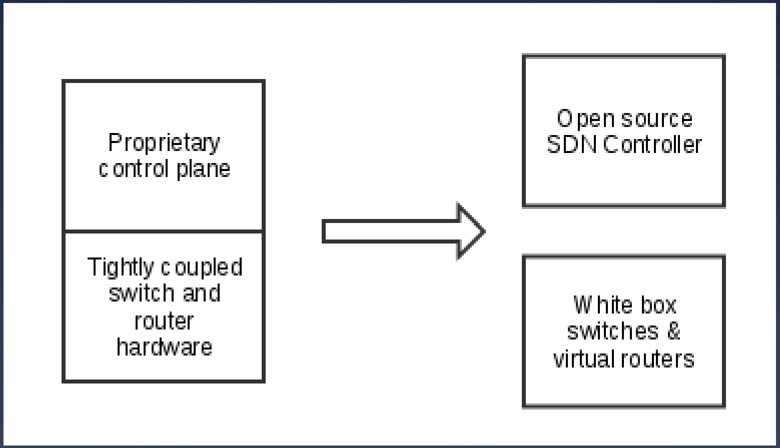
What is SDN?

“SDN is an approach to computer networking that allows network administrators to manage network services through abstraction of higher-level functionality. This is done by decoupling the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the selected destination (the data plane).”

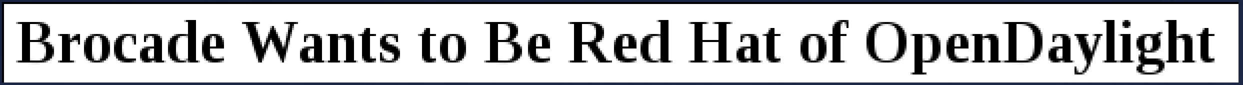
- The Wikipedia hive mind

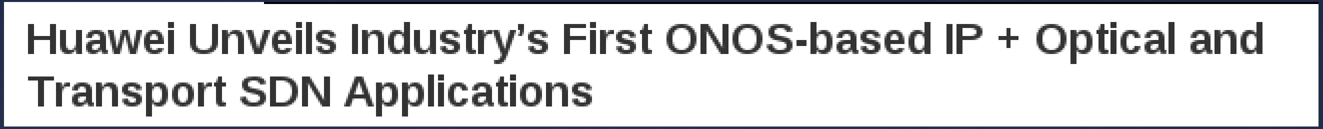

Proprietary UNIX OS
Tightly coupled CPU architecture
Linux/BSD
x86 commodity hardware

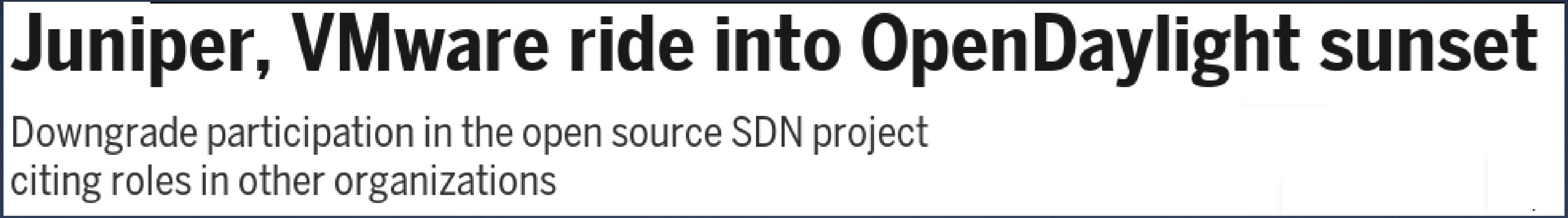
Digital UNIX
AIX
hp UX
ORACLE SOLARIS


Proprietary control plane
Tightly coupled switch and router hardware
Open source SDN Controller
White box switches & virtual routers

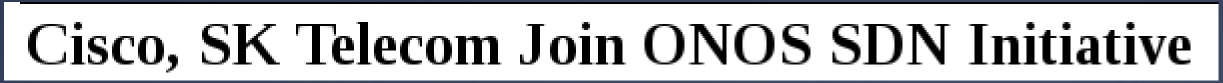
BROCADE
HUAWEI
JUNIPER NETWORKS
CISCO

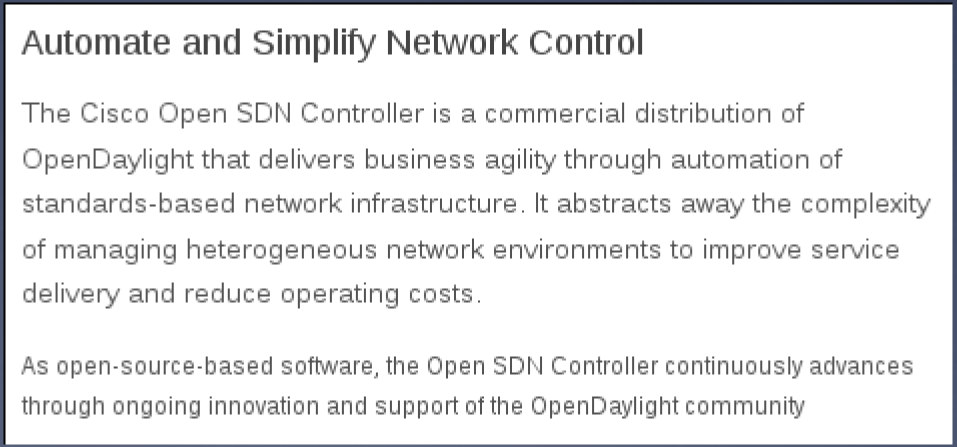

Brocade Wants to Be Red Hat of OpenDaylight


Huawei Unveils Industry's First ONOS-based IP + Optical and Transport SDN Applications


Juniper, VMware ride into OpenDaylight sunset

Downgrade participation in the open source SDN project
citing roles in other organizations


Cisco, SK Telecom Join ONOS SDN Initiative


Automate and Simplify Network Control

The Cisco Open SDN Controller is a commercial distribution of OpenDaylight that delivers business agility through automation of standards-based network infrastructure. It abstracts away the complexity of managing heterogeneous network environments to improve service delivery and reduce operating costs.

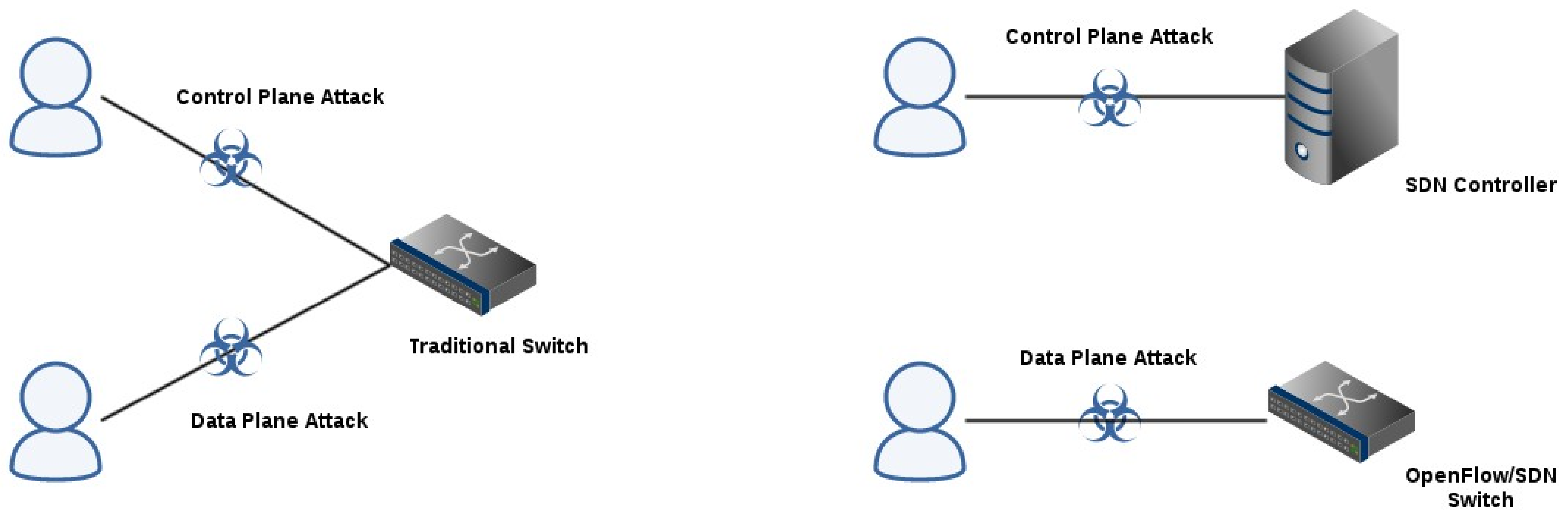
As open-source-based software, the Open SDN Controller continuously advances through ongoing innovation and support of the OpenDaylight community

SDN attack surface

SDN Attack Surface

- Traditional networks conflate the control and data planes on a physical device
- Software-defined networks factor the control plane out to a SDN controller.
- The controller uses a protocol such as OpenFlow to control switches, which are now only responsible for handling the data plane
- Advantage: easily segregate the control plane network from the production data network
- Disadvantage: the SDN controller's ability to control an entire network makes it a very high value target

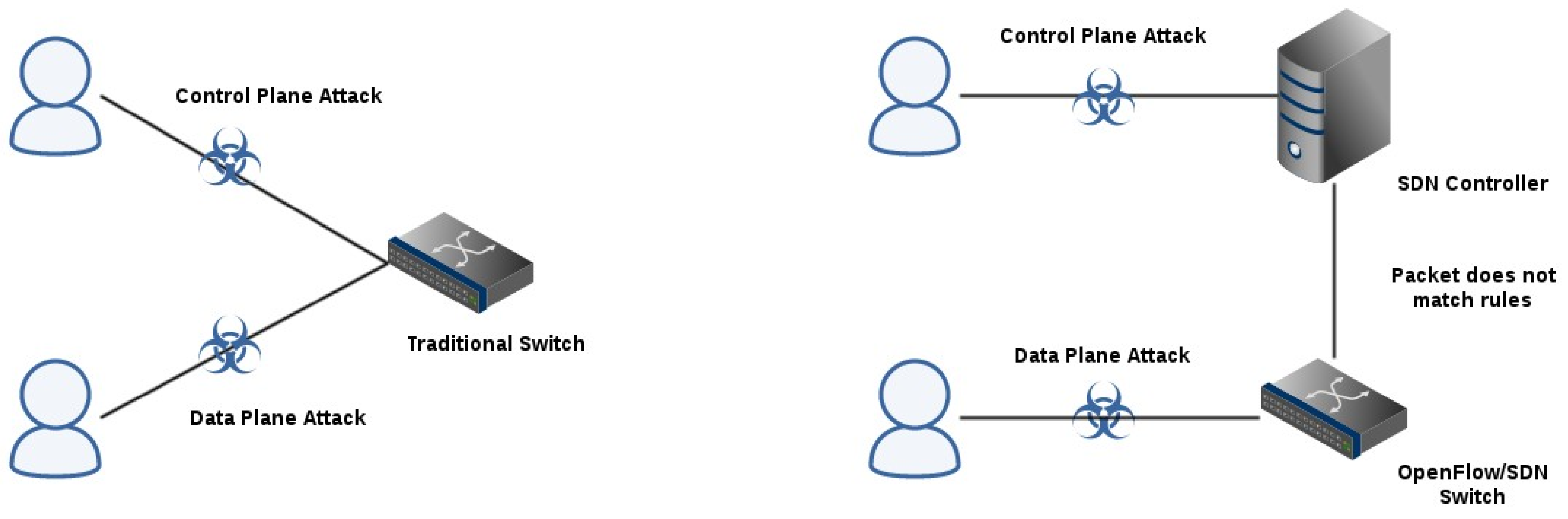
SDN Attack Surface



SDN Attack Surface

- SDN controllers are also exposed via the data plane
- When an OpenFlow switch encounters a packet that does not match any forwarding rules, it passes this packet to the controller for advice.
- As a result, it is possible for an attacker who is simply able to send data through an SDN switch to exploit a vulnerability on the controller.
- Switches out of scope for this presentation. See Gregory Pickett's BH 2015 talk if you're interested.

SDN Attack Surface



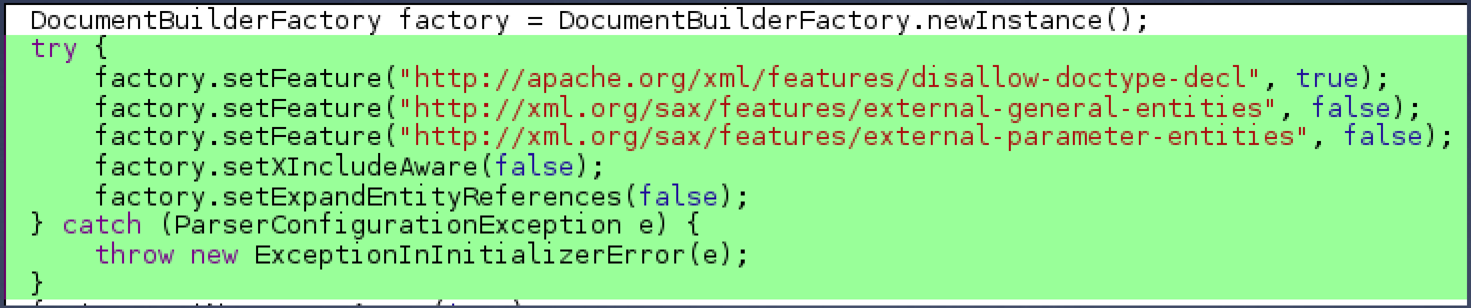
Recent SDN vulnerabilities

SDN Controller Vulns

- There are many competing SDN controller implementations
- The two most prominent ones are open source, written in Java/OSGi, and backed by many large vendors
- OpenDaylight/ODL (Linux Foundation)
- ONOS (lots of Chinese backing: telcos, Huawei, etc.)

Netconf CVE-2014-5035

- ODL Netconf API processes user-supplied XML (also restconf)
- Example vuln code: controller / opendaylight/netconf/netconf-util/src/main/java/org/opendaylight/controller/netconf/util/xml/XmlUtil.java



```java
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
try {
    factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
    factory.setFeature("http://xml.org/sax/features/external-general-entities", false);
    factory.setFeature("http://xml.org/sax/features/external-parameter-entities", false);
    factory.setXIncludeAware(false);
    factory.setExpandEntityReferences(false);
} catch (ParserConfigurationException e) {
    throw new ExceptionInInitializerError(e);
}
```

- Demo…

Topology spoofing via host tracking CVE-2015-1611

- Most SDN controllers include host tracking, allowing hosts to migrate between different physical locations in the network.
- Host tracking is based on monitoring of Packet-In messages, and does not require any validation, authentication, or authorization.
- An attacker can impersonate a host and make the SDN controller believe it has migrated to a physical network location controlled by the attacker.

Topology spoofing via host tracking CVE-2015-1611

- For an attacker to exploit this flaw, they only need to be able to send malicious messages through a switch controlled by an SDN controller (i.e. data plane)
- The only pre-requisite is that the attacker must know the MAC address of the target host. For more details on this flaw, see: http://www.internetsociety.org/sites/default/files/10_4_2.pdf

DoS in ONOS packet deserializer CVE-2015-1166

- When an OpenFlow switch encounters a packet that does not match any forwarding rules, it passes this packet to the controller for advice.
- It was found that the packet deserializers in ONOS would throw exceptions when handling malformed, truncated, or maliciously-crafted packets.
- The exceptions were not caught and handled, which would result in the relevant switch being disconnected because an exception occurred in an I/O thread.
- Demo...

Defensive technologies

Topoguard

- The same research team that reported the topology spoofing flaw developed topoguard to mitigate it
- Verifies the conditions of a host migration.
- A legitimate host migration would involve a Port Down signal before the host migration finishes. It would also mean that the host would be unreachable at its old physical network location after the migration is complete.
- Currently tightly coupled to the Floodlight controller

Security-mode ONOS

- A new feature targeting the upcoming ONOS 'Cardinal' release.
- Effectively a mandatory access control (MAC) implementation for ONOS applications
- Applications can be constrained by a policy dictating which actions they are permitted to perform.
- A vulnerability in an ONOS application could not be exploited to perform actions that are not permitted by security-mode ONOS. This is similar to the protection SELinux provides for applications running on Linux systems.

Security response best practices

Open Source Security Response

- All information public
- Not just source code: bug trackers, mailing lists, etc.
- Security requires the opposite approach: information must be kept private until patches are available
- How do you handle this in the context of an open source project?
- A dedicated security team with a documented process

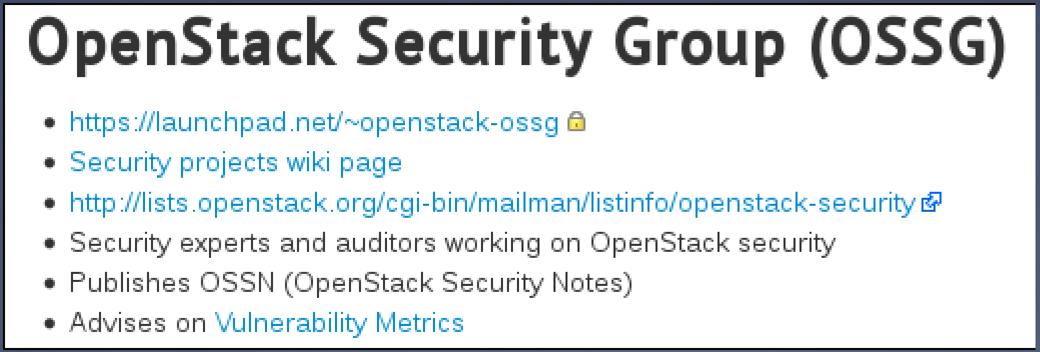
Open Source Security Response

- Dedicated mechanism for reporting security issues, separate to normal bugs
- Dedicated team with a documented process for responding to these reports
- Ability to quickly build a patch asynchronous to normal release schedules
- Clear documentation of the issue in an advisory, including references to patch commits (advantage of open source)
- More transparent than proprietary vendors (FireEye, Oracle…)

Secure engineering best practices

Open Source Secure Engineering

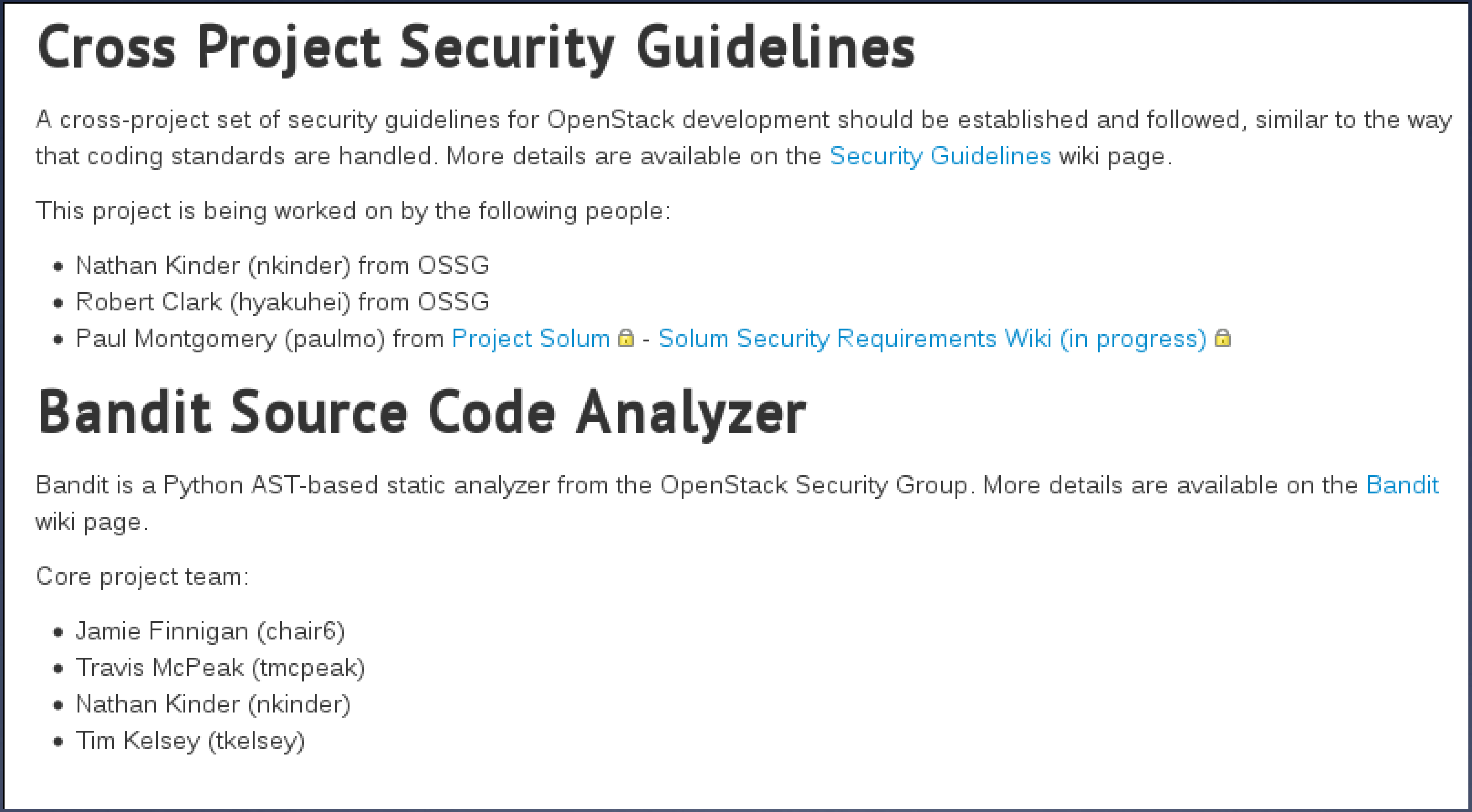
- No well established best practices
- Few good examples in the open source world. Proprietary software currently does this better, e.g./ microsoft's SDLC.
- OpenStack is one good example
- Separate VMT and OSSG teams



OpenStack Security Group (OSSG)

- https://launchpad.net/~openstack-ossg
- Security projects wiki page
- http://lists.openstack.org/cgi-bin/mailman/listinfo/openstack-security
- Security experts and auditors working on OpenStack security
- Publishes OSSN (OpenStack Security Notes)
- Advises on Vulnerability Metrics

Open Source Secure Engineering



Cross Project Security Guidelines

A cross-project set of security guidelines for OpenStack development should be established and followed, similar to the way that coding standards are handled. More details are available on the Security Guidelines wiki page.

This project is being worked on by the following people:

- Nathan Kinder (nkinder) from OSSG
- Robert Clark (hyakuhei) from OSSG
- Paul Montgomery (paulmo) from Project Solum - Solum Security Requirements Wiki (in progress)

Bandit Source Code Analyzer

Bandit is a Python AST-based static analyzer from the OpenStack Security Group. More details are available on the Bandit wiki page.

Core project team:

- Jamie Finnigan (chair6)
- Travis McPeak (tmcpeak)
- Nathan Kinder (nkinder)
- Tim Kelsey (tkelsey)

Open Source Secure Engineering

- Secure development guidelines (relies on developers to implement)
- Developer training (I just did this for everyone in the room, but it is“expensive” and difficult to roll out in a virtual environment)
- Automated QE/CI jobs to catch issues and enforce standards, e.g. via static analysis

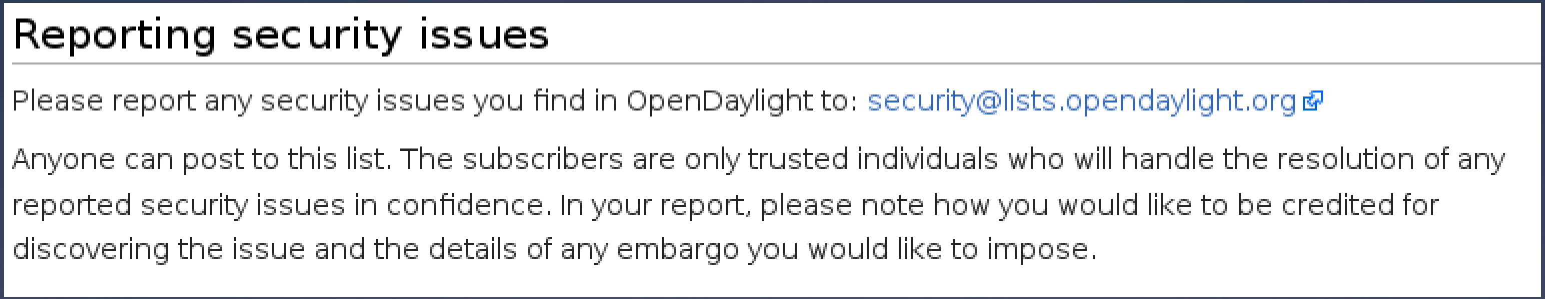
Find Security Bugs

- Static analysis with 56 bug patterns
- http://h3xstream.github.io/find-sec-bugs/

ODL: Current security status

ODL: Security Response

- Security reporting mechanism
- Dedicated team with a private mailing list and basic process for handling issues

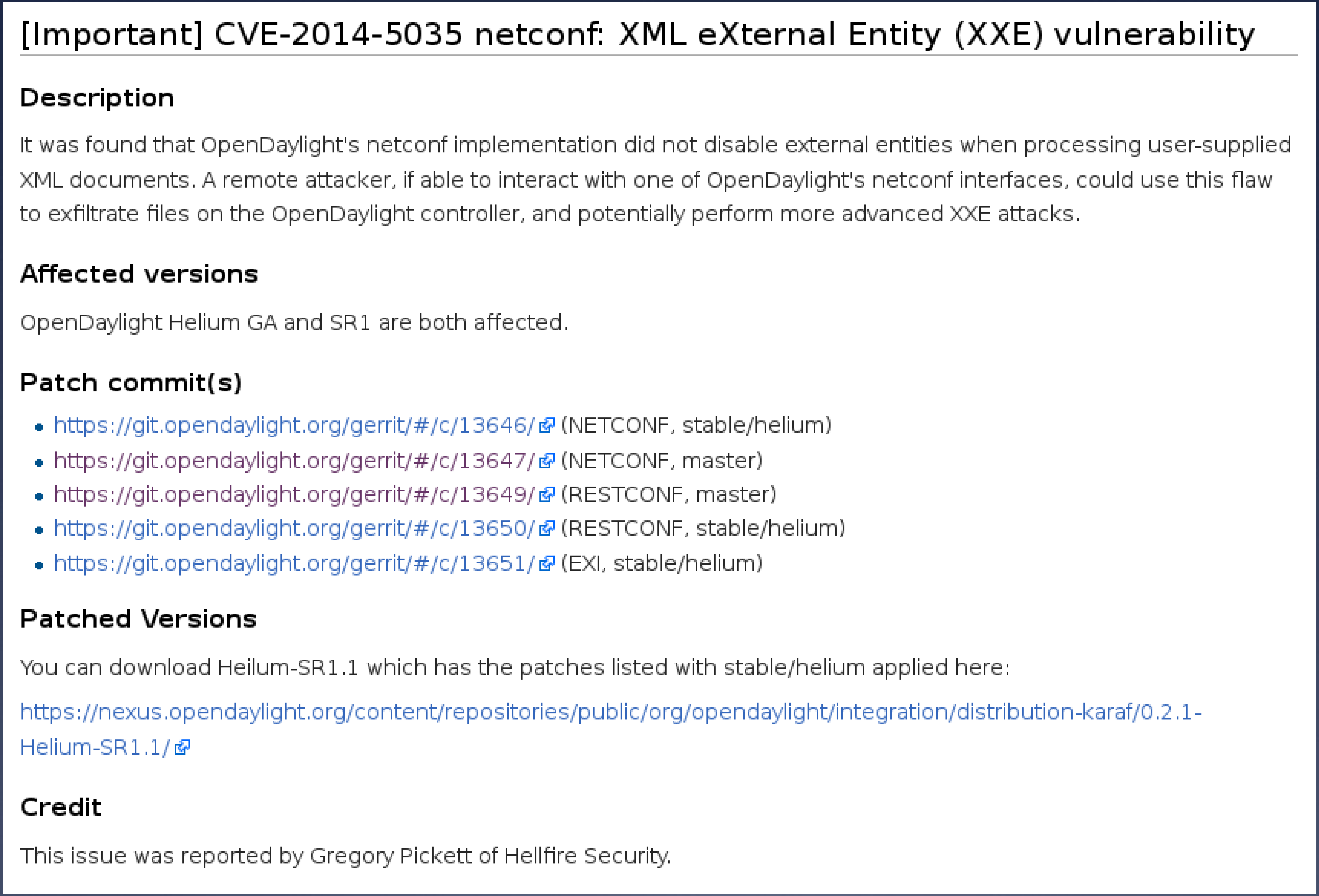


Reporting security issues

Please report any security issues you find in OpenDaylight to: security@lists.opendaylight.org

Anyone can post to this list. The subscribers are only trusted individuals who will handle the resolution of any reported security issues in confidence. In your report, please note how you would like to be credited for discovering the issue and the details of any embargo you would like to impose.

- Security advisories page



[Important] CVE-2014-5035 netconf: XML eXternal Entity (XXE) vulnerability

Description

It was found that OpenDaylight's netconf implementation did not disable external entities when processing user-supplied XML documents. A remote attacker, if able to interact with one of OpenDaylight's netconf interfaces, could use this flaw to exfiltrate files on the OpenDaylight controller, and potentially perform more advanced XXE attacks.

Affected versions

OpenDaylight Helium GA and SR1 are both affected.

Patch commit(s)

- https://git.opendaylight.org/gerrit/#/c/13646/ (NETCONF, stable/helium)
- https://git.opendaylight.org/gerrit/#/c/13647/ (NETCONF, master)
- https://git.opendaylight.org/gerrit/#/c/13649/ (RESTCONF, master)
- https://git.opendaylight.org/gerrit/#/c/13650/ (RESTCONF, stable/helium)
- https://git.opendaylight.org/gerrit/#/c/13651/ (EXI, stable/helium)

Patched Versions

You can download Heilum-SR1.1 which has the patches listed with stable/helium applied here:

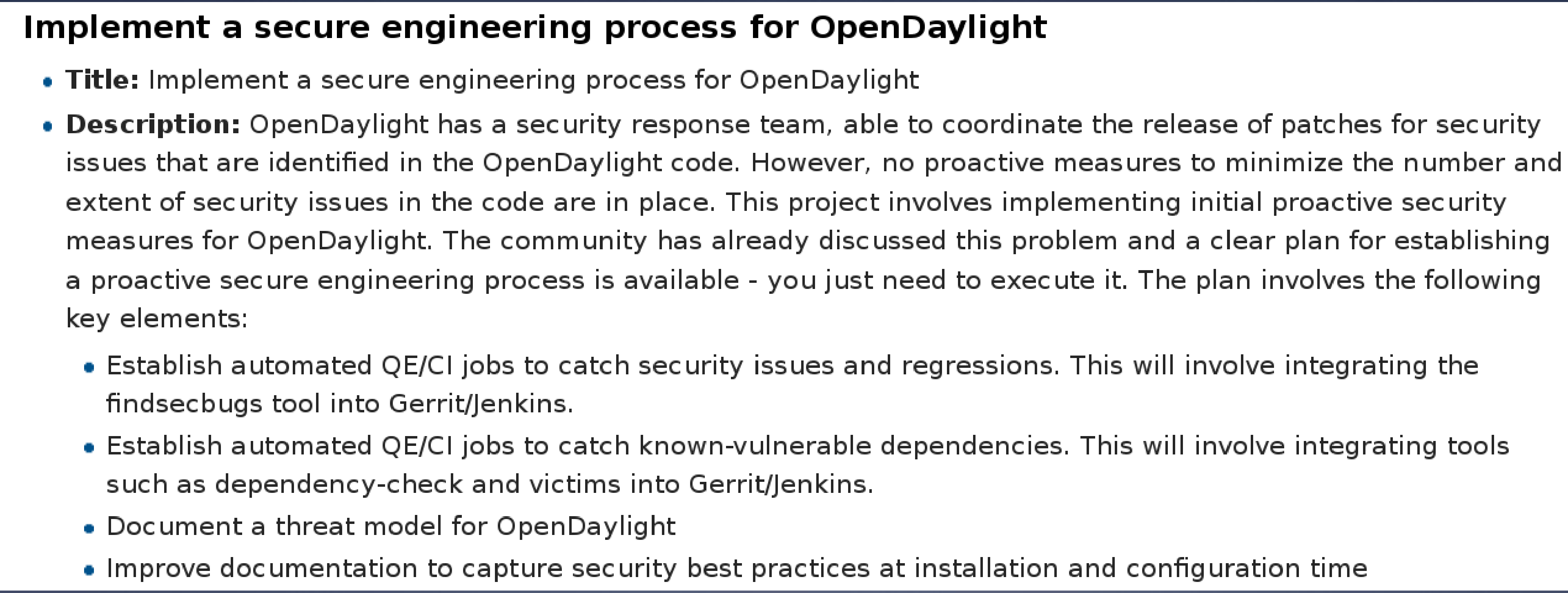
https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.2.1-Helium-SR1.1/

Credit

This issue was reported by Gregory Pickett of Hellfire Security.

ODL: Secure Engineering

- Great analysis performed in May 2014, but no action on fixing things. Cue the ODL summer internship program.



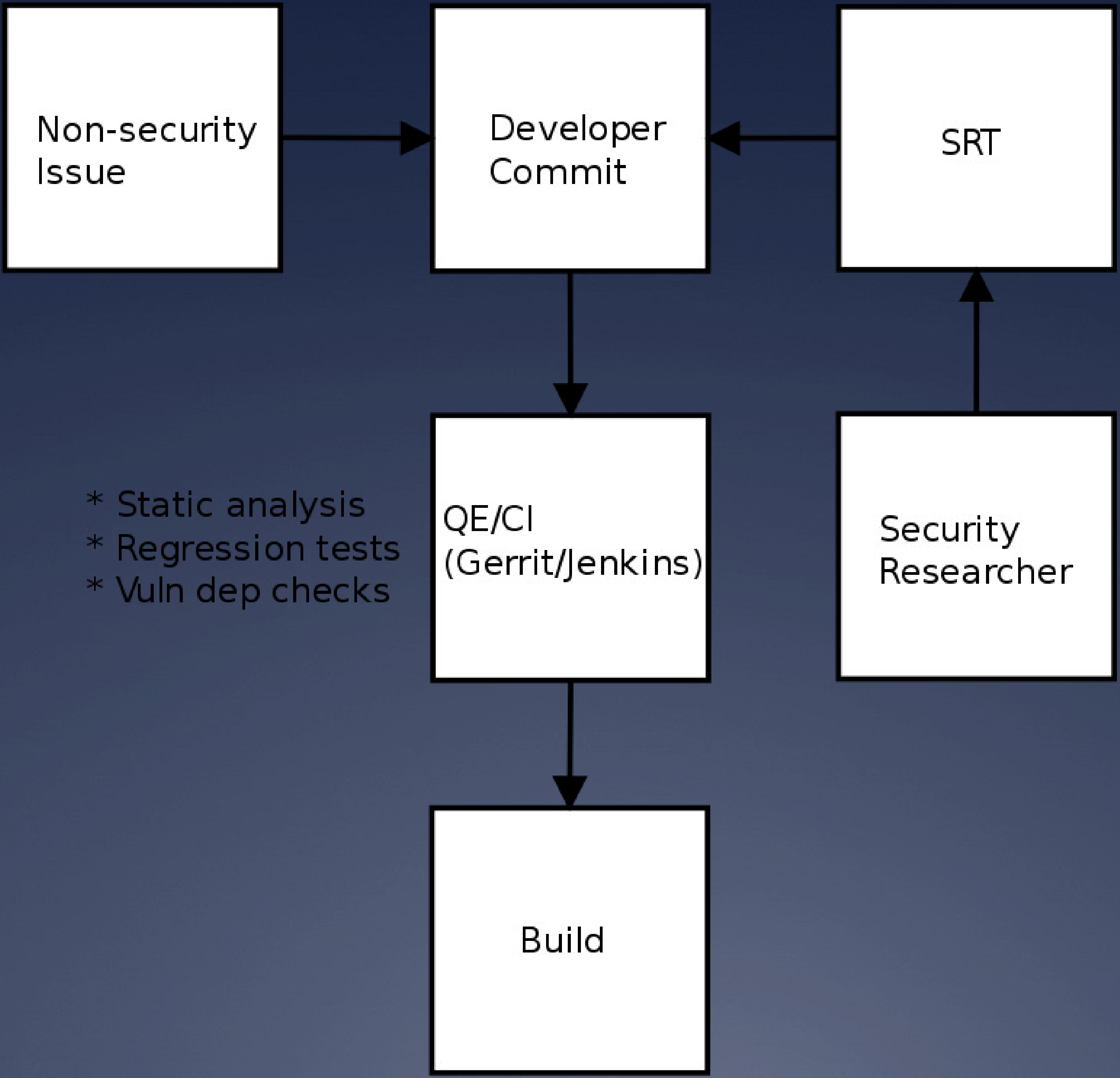
Implement a secure engineering process for OpenDaylight

- **Title:** Implement a secure engineering process for OpenDaylight
- **Description:** OpenDaylight has a security response team, able to coordinate the release of patches for security issues that are identified in the OpenDaylight code. However, no proactive measures to minimize the number and extent of security issues in the code are in place. This project involves implementing initial proactive security measures for OpenDaylight. The community has already discussed this problem and a clear plan for establishing a proactive secure engineering process is available - you just need to execute it. The plan involves the following key elements:
  - Establish automated QE/CI jobs to catch security issues and regressions. This will involve integrating the findsecbugs tool into Gerrit/Jenkins.
  - Establish automated QE/CI jobs to catch known-vulnerable dependencies. This will involve integrating tools such as dependency-check and victims into Gerrit/Jenkins.
  - Document a threat model for OpenDaylight
  - Improve documentation to capture security best practices at installation and configuration time

ODL: Security vision

ODL: Security Vision

- Industry leading secure engineering function
- Security docs (e.g. best practice install guide)
- Developer training as part of committer onboarding
- Automated QE/CI jobs to catch issues and regressions
- No reliance on documented secure coding standard (automate any standards in QE/CI jobs)


Non-security Issue
Developer Commit
SRT
* Static analysis
* Regression tests
* Vuln dep checks
QE/CI (Gerrit/Jenkins)
Security Researcher
Build

Next steps

Next Steps

- More research into data plane → control plane attacks
- Greater focus from the offensive security community as a whole – so far only Pickett and I seem to be looking
- Can we get a decent implementation **not** written in Java?
- Big vendors: please give at least one single fuck!

Questions?