HACKING THE BRAIN

Customize Evil Protocol to Pwn an SDN Controller

Feng Xiao, Ph.D. student at PennState, Cyber Security Lab
Jianwei Huang, Researcher at Wuhan University
Peng Liu, Professor at PennState, Cyber Security Lab
Software-Defined Networking (SDN) is an emerging architecture that decouples the network control and forwarding functions.
What’s SDN Like Today?

Who are contributing?

- More than 15 popular controllers.
- More than 1700 open source SDN projects.

Who are using?

- Data Center
- Telecom
- Enterprise
- …
## Attack Objectives in SDN

<table>
<thead>
<tr>
<th>Objective</th>
<th>Reference</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Congest control channel</td>
<td>Control plane saturation attack</td>
<td>Denial of Service</td>
</tr>
<tr>
<td>Terminate/Disrupt network services</td>
<td>State manipulation attack</td>
<td>Denial of Service</td>
</tr>
<tr>
<td>Steal confidential configuration</td>
<td>New</td>
<td>Data leakage</td>
</tr>
<tr>
<td>Probe network information</td>
<td>New</td>
<td>Data leakage</td>
</tr>
<tr>
<td>Install flow rules</td>
<td>New</td>
<td>Network manipulation</td>
</tr>
<tr>
<td>Fabricate links or hosts</td>
<td>Topology poisoning attack</td>
<td>Network manipulation</td>
</tr>
<tr>
<td>Distort network service results</td>
<td>New</td>
<td>Network manipulation</td>
</tr>
<tr>
<td>Disconnect network elements</td>
<td>New</td>
<td>Network manipulation</td>
</tr>
<tr>
<td>Install malicious SDN applications</td>
<td>New</td>
<td>Network manipulation</td>
</tr>
</tbody>
</table>

![Diagram showing SDN architecture with APPs, Controller, Control Plane, and Data Plane](attachment://sdn_diagram.png)
Pwn It Like A Hacker

Software-Defined Networks

Decoupled Control Plane and Data Plane

Controller
- Firewall
- Load-Balancing
- ...

Control Channel
- OpenFlow
- OVSDB
- ...

Infrastructure
- Switch
- Host
- ...
Pwn It Like A Hacker

Our Choice: Custom Attack

Controller
- Firewall
- Load-Balancing
- ...

Control Channel
- OpenFlow
- OVSDB
- ...

Infrastructure
- Switch
- Host
- ...
Custom Field (CF) in legitimate protocol interactions

- CF is controlled by data plane (hacker)
- CF will be processed by components in the controller
Custom Field (CF) in legitimate protocol interactions

- CF is controlled by data plane
- CF will be processed by components in the controller

CF results in a semantic gap between control plane and data plane
What Can It Cause?

- Execute Arbitrary SDN Commands
- Steal Confidential Data
- Crash/Disrupt Service
- Disable Network Function
- ...
Threat Model

We do NOT assume that hackers can have network access to SDN controllers or SDN applications.

Control channel is well protected by SSL/TLS.
Threat Model

We do NOT assume that hackers can have network access to SDN controllers or SDN applications.

Control channel is well protected by SSL/TLS.

A compromised host\(^1\) or switch\(^2\)

---

\(^1\) exploitable if the target network is configured with in-band control.
\(^2\) Switches are vulnerable to multiple remote attacks (e.g., Buffer Overflow[CVE-2016-2074]).
Challenges

Abusing Custom Field is not enough.

• Every Component runs in its separate context.
Challenges

Abusing Custom Field is not enough.

• Every Component runs in its separate context.
• Critical components are usually specially protected.
Multi-stage Exploitation

1: Toe-hold Stage
2: Harvest Stage
3: Chaining Stage
Multi-stage Exploitation

1: Toe-hold Stage
2: Harvest Stage
3: Chaining Stage
Multi-stage Exploitation

1: Toe-hold Stage
- Compromise a network device
- Exploit vulnerable Custom Fields
- Payload in Form 1

2: Harvest Stage
- Harvest more assets

3: Chaining Stage
- Link Discovery
- Routing
- Controller

Infrastructure

APP

Custom Crafted Protocol Message
Multi-stage Exploitation

1: Toe-hold Stage
2: Harvest Stage
3: Chaining Stage

- Subvert SDN Controller
- Harvest more assets
- Exploit vulnerable Custom Fields
- Compromise a network device

Payload in Form N
Payload in Form 1
Crafed Protocol Message

Controller
APP
Service
Routing
Link Discovery

Infrastructure

101 011
ONOS Remote Command Execution
pi@openvswitch:~$ # ONOS Controller is root@controller (192.168.1.111)
pi@openvswitch:~$
pi@openvswitch:~$ # Our compromised switch is this machine (192.168.1.108)
pi@openvswitch:~$
```java
private static final String COMMAND = "./bin/onos-node-diagnostics";
private static final String DIAGS = "/tmp/onos-node-diags.tar.gz";

private final Logger log = LoggerFactory.getLogger(getClass());

/**
 * Get tar.gz stream of node diagnostic information.
 * @return 200 OK with a tar.gz stream of diagnostic data
 */
@GET
@Produces(MediaType.APPLICATION_OCTET_STREAM)
public Response getDiagnostics() {
    try {
        execute(COMMAND);
        return ok(new FileInputStream(DIAGS)).build();
    }
```
Hack Something Real!
Hack Something Real!
# All users, groups, and roles entered in this file are available after Karaf startup
# and modifiable via the JAAS command group. These users reside in a JAAS domain
# with the name "karaf".
#
karaf = karaf, _g_ : admingroup
onos = rocks, _g_ : admingroup
onos1 = rocks, _g_ : admingroup
guest = guest, _g_ : guestgroup

_g_\: admingroup = group, admin, manager, viewer, webconsole
_g_\: guestgroup = group, viewer
public Collection<Alarm> translateToAlarm(DeviceId deviceId, InputStream message) {
    try {
        Collection<Alarm> alarms = new ArrayList<>();
        Document doc = createDocFromMessage(message);

        // parse date element value into long
        Node eventTime = doc.getElementsByTagName(EVENTTIME_TAGNAME).item(0);
        String date = eventTime.getTextContent();
        long timestamp = parseDate(date);
    }
}
// Extracts the ZIP stream into the specified directory.
private void extractZipArchive(File dir, InputStream stream) throws IOException {
    ZipInputStream zis = new ZipInputStream(stream);
    ZipEntry entry;
    while (entry = zis.getNextEntry() != null) {
        if (!entry.isDirectory()) {
            byte[] data = toByteArray(zis);
            zis.closeEntry();
            File file = new File(dir, entry.getName());
            createParentDirs(file);
            // write(data, file);
        }
    }
    zis.close();
}
Hack Something Real!

Path Traversal

Command Execution

Plaintext Key

XSS

XXE
Evaluation

5 popular SDN Controller

- Three open source projects (White-box)
- Two commercial products (Black-box)

54 apps

- Analyze 12 protocols
- Identify 476 dangerous function calls

18 zero-day vulnerabilities

- Construct 24 sophisticated exploit chains
Impact Analysis

Get System Shell (1 of them)

Execute Arbitray SDN Commands (5 of them)

Steal Confidential Data (7 of them)

Crash/Disrupt Service (11 of them)
<table>
<thead>
<tr>
<th>Controller</th>
<th>Bug#</th>
<th>Component Name</th>
<th>Stage</th>
<th>Vulnerability Description</th>
<th>Compatible Attack Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>T</td>
<td>H</td>
<td>C</td>
</tr>
<tr>
<td>ONOS</td>
<td>1</td>
<td>NETCONF</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Driver</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Device UI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Karaf</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>OVSDB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Core</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>YANG</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Floodlight</td>
<td>8</td>
<td>Switch UI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>RestServer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Forwarding</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Web</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OpenDaylight</td>
<td>12</td>
<td>SDNI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>VPNService</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>IoTDM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>HPE VAN</td>
<td>15</td>
<td>Monitor UI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>System Configuration</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SDNC</td>
<td>17</td>
<td>UI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Rest API</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

T: Toehold stage  H: Harvest stage  C: Chaining stage
1#: Command Execution  2#: Service Disruption  3#: Data Leakage
Researchers from Fraunhofer AISEC also discovered Bug#3.
Thanks!

Email : f3ixiao@gmail.com

Homepage: http://fxiao.me

Twitter: @f3ixiao