Penetration Test Report

for
Sitting Duck B.V.

V1.0
Amsterdam January 26th, 2015
60628081

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 FishInABarrel Web Application to Radically Open Security BV (hereafter “ROS”). Sitting Duck has made this request to better evaluate the security of the application and to identify application level vulnerabilities in order to see whether the FishInABarrel Web Application is ready, security-wise, for production deployment.

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 target:
• fishinabarrel.sittingduck.com

The penetration test was carried out from a black box perspective: no information regarding the system(s) tested was provided by Sitting Duck or FishInABarrel, although FishInABarrel did provide ROS with two test user accounts.
was provided by Sitting Duck or FishInABarrel, although FishInABarrel did provide ROS with two test user accounts.
The scope of the Sitting Duck penetration test was limited to the following target:
• fishinabarrel.sittingduck.com

The penetration test was carried out from a black box perspective: no information regarding the system(s) tested was provided by Sitting Duck or FishInABarrel, although FishInABarrel did provide ROS with two test user accounts.

1.3 Project objectives

The objective of the security assessment is to gain insight into the security of the host and the FishInABarrel Web Application.

1.4 Timeline

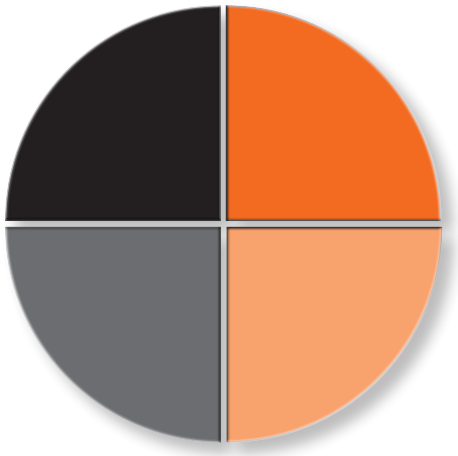
The FishInABarrel Security Audit took place between January 14 and January 16, 2015.

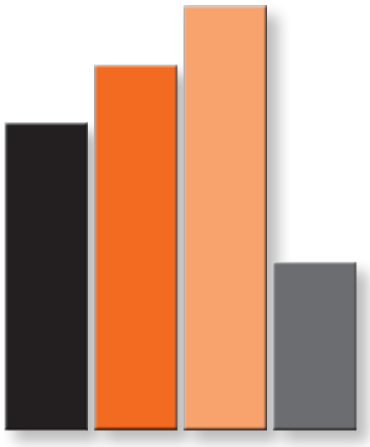
1.5 Results in a Nutshell

During this pentest, we found quite a number of different security problems – Cross-site Scripting (XSS) vulnerabilities, both stored and reflected, Cross-site Request Forgery (CSRF) vulnerabilities, information disclosures (multiple instances), and lack of brute force protection.
During this pentest, we found quite a number of different security problems – Cross-site Scripting (XSS) vulnerabilities, both stored and reflected, Cross-site Request Forgery (CSRF) vulnerabilities, information disclosures (multiple instances), and lack of brute force protection.was provided by Sitting Duck or FishInABarrel, although FishInABarrel did provide ROS with two test user accounts.was provided by Sitting Duck or FishInABarrel, although FishInABarrel
The penetration test was carried out from a black box perspective: no information regarding the system(s) tested was provided by Sitting Duck or FishInABarrel, although FishInABarrel did provide ROS with two test user accounts.

1.6 Summary of Findings

ID	Type	Description	Threat level
SID-001	Information Leak	The phpinfo() function of the PHP language is readable, resulting in a listing of all the runtime information of the environment, thus disclosing potentially valuable information to attackers.	Moderate
SID-002	XSS	A general description of the problem.	High
SID-003	XSS	A description of the problem.	Low





2 Methodology

2.1 Planning

Through automated scans we were able to gain the following information about the software and infrastructure.

Detailed scan output can be found in the sections below.

1. Reconnaissance

We attempted to gather as much information as possible about the target. Reconnaissance can take two forms: 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 be carried out to tailor the test to the discovered services.

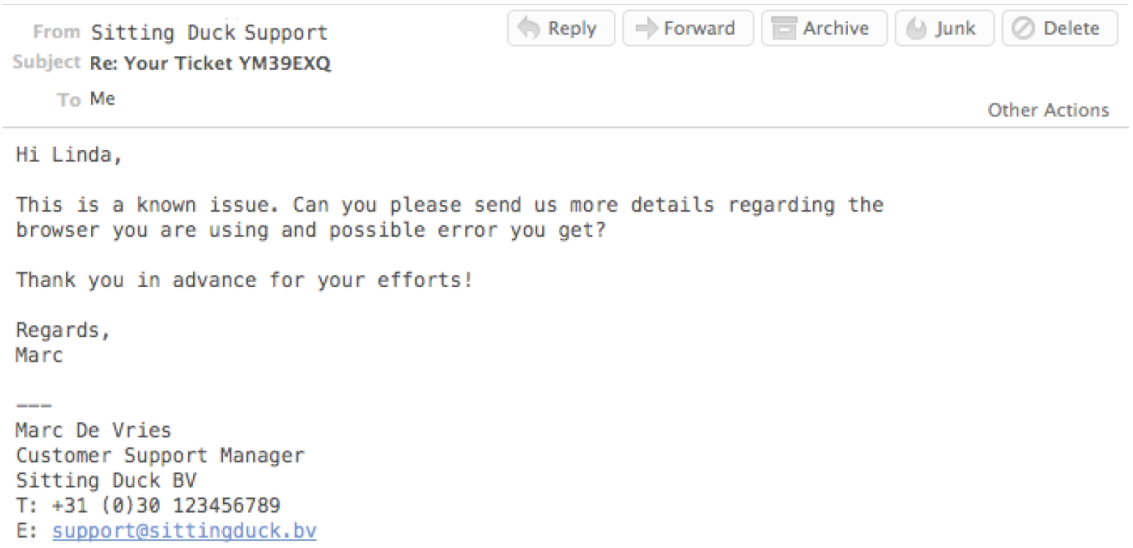
3. Scanning

Through the use of vulnerability scanners, all discovered hosts would be tested for vulnerabilities. The result would 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

Through the 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.

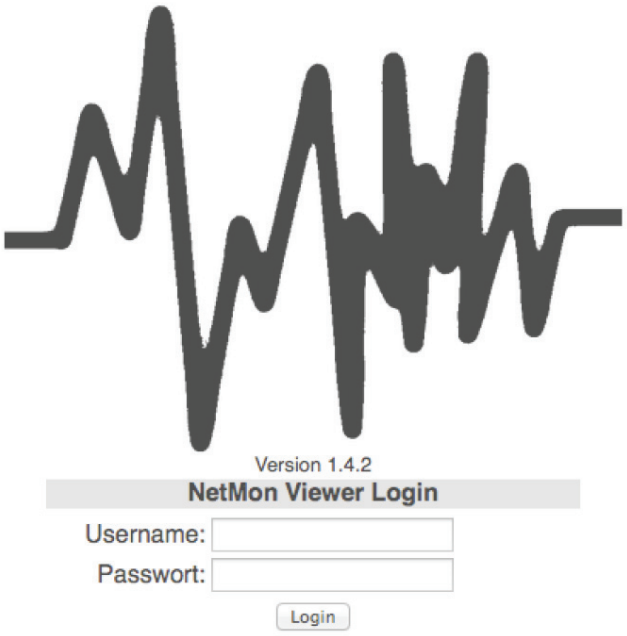
We used varied operating system fingerprinting tools to determine what hosts are alive on the network and more importantly what services and operating systems they are running. Research into these services would be carried out to tailor the test to the discovered services. We used varied operating system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



2.2 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.

3 Reconnaissance and Fingerprinting

Through automated scans we were able to gain the following information about the software and infrastructure. Detailed scan output can be found in the sections below.

Fingerprinted Information
Windows XP
Microsoft IIS 6.0
PHP 5.4.29
jQuery 1.7.2
Mailserver XYZ
FTPserver ABC

3.1 Automated Scans

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

- nmap – http://nmap.org
- skipfish - https://code.google.com/p/skipfish/
- sqlmap – http://sqlmap.org
- Wapiti – http://wapiti.sourceforge.net

Of these, only the output of nmap turned out to be useful; consequently only nmap and output will be discussed in this section.

3.2 nmap

Command:

$ nmap -vvvv -oA fishinabarrel.sittingduck.com_complete -sV -sC -A -p1-65535 -T5

Outcome:

Nmap scan report for fishinabarrel.sittingduck.com (10.10.10.1)

Starting Nmap 4.11 (http://www.insecure.org/nmap/) at 2013-11-11 15:43 EST Initiating ARP Ping Scan against 10.10.10.1 [1 port] at 15:43

The ARP Ping Scan took 0.01s to scan 1 total hosts.

Initiating SYN Stealth Scan against fishinabarrel.sittingduck.com (10.10.10.1)

[1680 ports] at 15:43 Discovered open port 22/tcp on 10.10.10.1

Discovered open port 80/tcp on 10.10.10.1

Discovered open port 8888/tcp on 10.10.10.1

Discovered open port 111/tcp on 10.10.10.1

Discovered open port 3306/tcp on 10.10.10.1


RADICALLY
OPEN
SECURITY