Information Security Workshop - Ex. 5

Eden Koveshi 316221746

March 27, 2019

This exercise intorduces two new components: *DLP*, and defense against a chosen attack: RCE via Hashicorp Consul Scripting Services.

Before using the firewall, run the following command in linux bash,from the working directory:

```
python ./module/firewall/proxy.py
```

Part I

Data Loss Prevention (DLP)

My DLP assigns the data a score, compares it to the average line length, and accepts/rejects wrt. some threshold

The idea was inspired by TAU's Information Security course, and some reading of McAfee DLP product documentation[1].

The Algorithm

The data is split into lines, and each line is given a score by the scoring function.

In the end, the data is classified code iff

$$\frac{avg.\,line\,score+1}{avg.\,line\,length+1} > threshold$$

The Scoring Function

The scoring function attempts to find three kinds of patterns:

- Regular expressions pre-computed, describing the C language
- Characters that have special meaning in C,repeat often in C, and not too often in natural language.High confidence characters
- Characters that have special meaning in C, repeat often in C but so in natural language. Those are low confidence characters.
- Common punctuation symbols that may appear in both C and text, also divided to high and low confidence. Those gain a very very small score.

- Lines that start or end with {, } gain extra score
- Lines that end with . lose score

The more"confidence" found in line, the higher the score it gets.
The score is given wrt. the line length, so it then can be compared.

Why RegEx?

Regular expressions are the most basic building blocks of programming languags, used in every compiler. They are fast and give quite a good indication.

Though they can't identify a language completely, they are an integral part of a programming lanuage.Thus, C file will probably contain many matches, while a text file won't, and thus they are a good indicator.

The advantage of using regular expressions over identifying symbols, is that regular expressions capture (at least partially) the structure of C language, and help differentiate random use of keywords from a real use of keywords.

The advantage of using regular expression over other solutions such as CFG parser, or ML classifying techniques is that they require heavy, time-consuming computations which a firewall can't afford to have, while regular expressions are easy to work with,and computed only once.

Thus, regular expressions is a good solution that satisfies both accuracy and perforemence.

Ideas for some of the expressions were taken from C language CFG.[2]

An interseting/useful fact: Regular expressions define structure.Using regular expressions lets us identify more C-based languages, such as C++,C# or Java (with less accuracy of course) due to their structure.This idea can be extended to support any other programming/scripting language.

Parameter Tuning

Using many random C and text files to tune the parameters, the final parameters are (l is the current line length):

```
reg exp match score: 6l
high conf match score: 1.4l
low conf match score: 0.3l
high conf symbol score: 0.015l
low conf symbol score: 0.007l
score for lines that start or end with {,}: 1.8l
score for lines that end with . : -2.1l
threshold: 1.4
```

POC

Running this on every c/txt file in my machine's root directory, the results are:
C code:

```
373 correct out of 382 total (97.6439790576%)
average inspection time is 0.91472581419 seconds
```

Text:

```
886 correct out of 1005 total (88.1592039801%)
average inspection time is 0.812968847408 seconds
```

of course text false positives are much less scarying :)

Integrating DLP within the proxy server

The DLP is used within the proxy server introduced in last exercise, in two ways:

• HTTP POST request, simply check header to recognize POST reuqest, and send the data to the DLP

• SMTP DATA request.When recieving a packet on port 25, the proxy sever checks whether it is a DATA packet or not, and if it is, sends the data over to the DLP.

To support SMTP, the appropriate redirections were added in *redirect_in,redirect_out* functions introduced in last exercise, and a new socket listening on port 2025 with forward port 25 added to the proxy server.

Part II
Defense Against Chosen Attack: RCE via Hashicorp Consul Services API

Introduction

Consul is a service mesh soultion by Hashicorp.

One of the things Consul offers is the Services API - the ability to run services, potentially from scripts, on the machine to make tests. Sounds crazy already.

This was originally intended for local use only, and Consul customers have been warned for years.

When many of their customers did not take their advise and enabled remote access to the service, they have been suffering from RCE attacks.

To get the attack to work, one needs:

1. A flaw in the whitelisting of scripts, which is not hard to find at probably most organizations
2. ACL disabled or compromised

Attack Prevention

To prevent this, Hashicorp released new configuration flags, which allow script files to be registered only by local machine.

My Firewall Solution

As I don't have a Consul server, nor an ACL token, I cannot verify that the server is compromised.

If it's not, then blocking isn't needed.

However, if it's not compromised, then scripts can be run only locally either way.

Thus, my solution is simply blocking packets with a register request (can be found in the Metasploit attack source code[3])

So any request coming from outside network will be blocked, and local requests won't pass through the firewall so they'll pass, and the solution is efficiently the same as Hashicorp's solution.

References

1. https://docs.mcafee.com/bundle/data-loss-prevention-11.0.400-product-guide-epolicy-orchestrator/page/GUID-F54FD4EF-E613-4ADB-8109-46A96BDFE4FC.html
2. https://www.cs.dartmouth.edu/~mckeeman/cs118/xcom/doc/Ccfg.html
3. https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/misc/consul_se