|  |  |
| --- | --- |
| C:\user-pjpf\Briefcase-SO\so\so-12-13\slides\logo-IST-Tecnico.JPG | INSTITUTO SUPERIOR TÉCNICO  Departamento de Engenharia Informática  Forensics Cyber Security  MEIC / METI 2018-2019 – 1st Semester |

Digital Forensics Report

**Authors**

**GROUP I – Network trace analysis**

## Indicate the complete URL of the original web request that led to the client being compromised

http://10.20.0111:8080/banking.htm

## What file type was requested in the final web request to the malicious server?

GIF file.

## What is the number of the first frame that indicates that the client has been compromised?

4722. It is very suspicious that the client closed every TCP connection and open only one: A TCP to the malicious server (10.20.0.111).

## At one point, the malicious server sends a malicious file to the client. What type of file it is?

Windows – Executable PE32 executable (DLL) (GUI) Intel 80386, for MS Windows

## What is the SHA1 hash of the malicious file?

9972727c9612cf77d3bd233e7c5a4f22f76e2c7a

## What vulnerable software has been exploited (in the following format, Firefox 3.5, Firefox 3.6, Word 2010, IE7, Safari2, Chrome2, AdobeReader, IE9)?

Microsoft Internet Explorer 6

## When the capture ends, is the client still connected to the malicious attacker?

Yes, there are no TCP RST or FIN ACK requests.

## Can you indicate the corresponding CVE security bulletin that covers the vulnerability that was exploited here (answer in form of CVE-$year-$number)?

CVE-2010-0249

## From the capture, it is clear that the attacker gets a certain form of access (i.e. the interface), what (type of) access does the attacker “get” on the client?

He gains the possibility of running any executable on the client’s device.

## What is the frame that indicates that something strange might be going on?

Frame no. 8, where we can see the start of a series of ping requests to IPs in order to check for open ports on machines.

## What tool is generating this traffic?

It's used the nmap tool.

## What does this frame constitute the beginning of?

Ping Scan (aka Ping Sweep): it scans the entire network searching for online hosts.

## A switch was removed from the command to improve the speed, what was this switch (just the letters, case-sensitive)?

-**sU** -> switch to disable UDP scan.

## What switch was added to the final scan (case-sensitive)?

**-A** -> flag that tries to discover, among other things the operating system version.

**GROUP II – Website fingerprinting over Tor**

# Indicate: (a) the selected feature set, and (b) the accuracy of your classifier using this feature set. Give an account of other features you have tried before converging into this feature set. Explain how you’ve modified the source code in order to compute your proposed feature set.

The main features used were: **number of cells** in each file, **number of outgoing packets** per file, **number of ingoing packets** per file, the **duration of communication** (timestamp of the last request), and the **positions of all** ingoing and outgoing packets.

More features were used, but they didn´t increased our accuracy, namely:

* Fraction of outgoing requests.
* Fraction of ingoing requests.
* Extract all sequences of outcoming packets and with these values we applied some statistic knowledge (mean, standard deviation, median, 50th percentile).
* Do the same for the ingoing packets.
* Calculate the number of requests per second.
* Register the number of requests done in each second and apply some statistic knowledge (mean, standard deviation, median, 50th percentile).
* Maximum number of successive 1s.
* Maximum number of successive –1s.
* Number of bursts (n successive ingoing/outgoing requests), where n >1, n > 4, n > 10, n > 20.
* We divided our sizes vector in chunks of several sizes (10,20,30,50) without overlapping. For each chunk we counted the number of outgoing requests (at this point we started discarding a bit the ingoing requests because they were a lot more than the outgoing ones) and applied some statistical knowledge as presented before.
* Applied some statistical knowledge on the vectors with positions of outgoing and ingoing requests (each vector stores one type of requests).
* Register the direction of first 100 packets (if less than 100 packets, padded with 0).

The modification in the source was done exclusively in the file ‘fextractor.py’. We just added these features to the ‘features’ array.

## Can you spot a trend in the accuracy of the classifier as k increases? Explain the variations observed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | 1 | 2 | 5 | 10 | 15 |
| ACC | 0.912 | 0.869 | 0.797 | 0.715 | 0.653 |

**Table 1**

The accuracy decreases with the increase of k.

This can be easily explained. The classifier provided in this lab assigns the testing point to the non-monitored class, if any of k point disagrees. If we increase k, the probability of any k point disagrees also increases, resulting in a failed attribution. The traditional knn, when some k point disagrees, it is used a quorum algorithm based on the distance.

## Can you spot any trend in the TPR/FPR trade-off alongside the variation of k and nm? Justify.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| nm% | 1 | 2 | 5 | 10 | 15 |
| 0.1 | 1.0/0.085 | 0.889/0.043 | 0.782/0.013 | 0.685/0.007 | 0.605/0.0016 |
| 0.2 | 1.0/0.100 | 0.871/0.041 | 0.753/0.014 | 0.645/0.009 | 0.569/0.0038 |
| 0.4 | 1.0/0.109 | 0.850/0.044 | 0.720/0.013 | 0.603/0.007 | 0.521/0.0044 |
| 0.6 | 1.0/0.118 | 0.837/0.051 | 0.689/0.012 | 0.572/0.007 | 0.466/0.0055 |
| 0.8 | 1.0/0.127 | 0.825/0.053 | 0.670/0.015 | 0.532/0.007 | 0.426/0.0055 |

**Table 2**

Increasing k decreases TPR and FPR. This is explained exactly the same way as in 3.1. With this classifier all k nodes need to agree, if any one doesn’t, then the classifier doesn’t return a positive classification (See **Chart 1**), decreasing TPR and FPR. FPR decreases because in the case of k nodes don’t agree the page will be classified as non-monitored, so the case in which we blame an innocent has a reduced probability of occurring.

Increasing the percentage of non-monitored pages in training set decreases TPR and increases FPR. TPR decreasing is also easily explained. **Increasing %nm** increases the probability of existing neighbors that are nm, so the probability of having neighbors that disagree are higher, leading us (again) to the problem of considering a monitored page as non-monitored resulting in a negative classification. We found out that increasing FPR (See **Chart 2**) doesn’t make much sense. If we have more non-monitored pages, it is expected that the number of non-monitored pages classified as monitored pages reduces. We think that this happened because we are talking of really low values, if we test a lot of times it would eventually stabilize and then we would extract the real value of FPR.

## According to your results in Table 2, which configuration leads to a more successful attack? Do you think an attack with such a TPR / FPR trade-off may be effective in practice? Justify.

It depends on what is intended by the attacker. In this answer we will assume that our FPR decreases with the increasing of %nm, because it is what makes sense.

A successful attack is an attack that have a big TPR. That happens when we have a lower k and a lower %nm. This attack might not be effective in practice because FPR is relatively high, so we have for instance a high probability of blaming an innocent for accessing a webpage distributing child-pornography. It would probably be preferable to decrease a bit the TPR but also decreases FPR, off course the proportions in which we decrease FPR are higher that the proportions in which we decrease TPR.

Attachments

Chart 1: True positive rate by k and test set size

Chart 2: False positive rate by k and test set size