Group LPL

23-Group-<LPL>https://drive.google.com/drive/folders/1tuphqNgWo8ceMXlKj8oZu_r19M_fDos-

Name	SSN	Email	Mobile (optional)
Qun Lou		qlou005@ucr.edu	9515738020
Tingting Pei		tpei004@ucr.edu	9515584690
Wanfang Li		wli245@ucr.edu	9515076758

Name	Project Total	Project Dev	Project Comm	Attendance	In-person	Reading
Qun Lou	45%	50%	25%	20	14	80%
Tingting Pei	30%	30%	50%	20	20	90%
Wanfang Li	25%	20%	25%	20	16	90%
Total:	100%	100%	100%	-----	---	---

Tingting Pei: I did the adaboosting part with two methods and did a big part of communication about the project.

Qun Lou: Read papers and conclude possible solutions; Collect data sets and generate new features; Build basic OCC models; Implement real-time detection; Test on C2s and analyse.

Wanfang Li: OCC models and optimization, also implement the adaboosting with average rate. Reading papers and write document.

A Real-Time Detecting System for DNS-Based C2 Channel through One-Class-Classification Methods

Group LPL

Qun Lou
Department of CS
UCR
Riverside, California
qlou005@ucr.edu

Tingting Pei
Department of Computer Engineering
University of California, Riverside
Riverside, California
tpei004@ucr.edu

Wanfang Li
Department of CE
UCR
Riverside,California
zwang560@ucr.edu

Abstract

Nowadays, an increasing number of networks constantly suffer the threat of valuable and sensitive data being stolen by cyber-attackers. As is known, attackers are exploiting the Domain Name System (DNS) channel for exfiltrating data as well as maintaining tunneled command and control communications for malware. This is because DNS traffic is safe in IP level and allowed by many IDS, thus, providing a covert channel for attackers to encode low volumes of data without fear of detection.

Our project develops and evaluates a real-time mechanism for detecting DNS Command & Control (C2) channel only in need of the Fully Qualified Domain Name (FQDN) through One-Class Classification (OCC) methods. In the first place, we feed our OCC models with hyper features generated from FQDN and use AdaBoosting to optimize. In the second place, we implement a real-time detection system with an optional whitelist on bloom-filter. Consequently, our program can reach a 99.84% accuracy for detecting the most popular type of DNS C2 among all the 3 types.

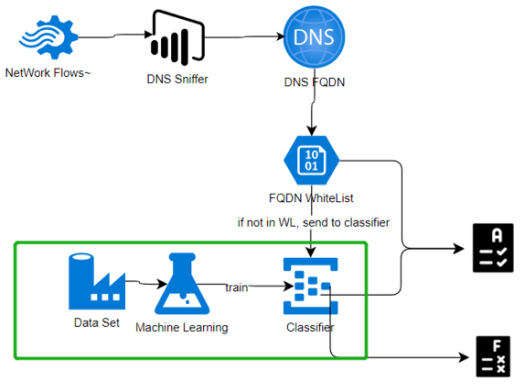
Innovations: 1)Discuss different types of existing DNS C2; 2)Apply OCC methods into DNS C2 detection and Evaluate; 3)Provide an open-source detecting system. Github

Language: 100% in Python; Lines of Code: about 700;

1 INTRODUCTION

1.1 Problem

Given a FQDN, train a model to tell whether it belongs to DNS C2's query or not. Moreover, implement a real-time detection system using that model.



1.2 **Previous Work**

M. Aiello, M. Mongelli and G. Papaleo[1] uses the content of the entire DNS database to deal with DNS tunnel detection through a simple supervised learning algorithm. They analyze secondary and tertiary DNS domains and focus on detecting a small part of malicious data hidden by regular DNS communication. This article finally got a good result through replicating a single test on consecutive samples over time and making a global decision through a majority voting scheme.

S. Sayadi, T. Abbes and A. Bouhoula[2] proposes a method based on the basic monitoring protocol "Internet Control Message Protocol" (ICMP) to monitor and detect the existence of hidden channels that can avoid basic security systems like firewalls. This paper processes network traffic through a series of simple or complex verification methods. These methods have a good performance on detecting this kind of malicious traffic.

IoTDS (Internet of Things Detection System) proposed by Bezerra, Vitor Hugo et al.[3] to detect botnets in IoT devices to protect the security of the Internet of Things. The classifier it uses does not require a manual labeling process and only models legitimate device behavior to detect deviations. It prevents IoT devices from being overloaded due to training activities and does not use network traffic data. The results show that the proposed system has a good detection effect on different botnets.

M. Aiello,M. Mongelli,G. Papaleo introduces a machine learning method for detecting DNS tunnels. This algorithm detects DNS intruders existing through aggregation-based monitoring. The advantage of this algorithm is that it can prevent packet inspection to protect privacy and scalability. It also overcomes the limitations of traditional classifiers by replicating a single detection on consecutive samples over time and making a global decision through a majority voting scheme.

In general, OCC methods have been widely used in anomaly detection and obtained beautiful results, but seem not for DNS C2, which motivates us to apply OCC. Also, some studies used the UNSW15 dataset for DNS C2, whose DNS C2 flows are generated by a single tool, Idoine. In contrast, we use a new dataset generated by 5 different tools. Indeed, we learn a lot for generating features from FQDN, and we also point out some new features like “entropy” and “average length of labels”, which are useful in further training.

1.3 **Solution**

OCC implement:We use 20000 records with 10000 normal data and 10000 abnormal data for fitting the model.And then We use 15000 records for testing the model, which has 10000 normal data and 5000 abnormal data. And we output the correctness rate every 3000 testing to inspect the process. In the model parameter part, we first set all the parameter by default. And we get the rough result with this model.

Optimization: After we took a deep look at the OCC models, we found that contamination this parameter in the most important parameter for the model we use. It is the amount of contamination of the data set, Used when fitting to define the threshold on the scores of the samples. This will tell the model how much portion of the data will be seen as abnormal, and be excluded from the Elliptic outliner. For the Insolation forest, the decrease the value of n_estimatorsint which is the number of base estimators in the ensemble. Its default value is 100, and for efficiency we set it to 10. For the same reason, we decrease the number of base estimators in the ensemble which is max_samples to 40. For One class SVM, we have tried other kernel but perform not good whose testing rate is below 50%, so we still choose rbf as our model. And the most important value in oneclassSVM is nu which is an upper bound on the fraction of training errors and a lower

bound of the fraction of support vectors. For this model its best value is 0.3 and we set it and get a better performance.

AdaBoosting:AdaBoosting is an optimization algorithm which is based on decision trees. In this algorithm, the classifier will fit the first algorithm with maximum accuracy. The second one needs to fix the errors of the first one, filter out the errors and enlarge them. The third fixes the previous error and goes on until the number of algorithms reaches the pre-specified value, and then combines these algorithms with weight. I execute the algorithm with methods. One is more complicated and has higher accuracy. The other one uses sklearn which is simpler in operation.

Real-time Detection:

Python API `scapy.layers.dns` to capture and parse DNS packets;

An optional whitelist saved in bloom filter for a high efficiency;

Python API `joblib` to load and run our well-trained models.

1.4 **Key Results**

OCCs implement and optimization: There different OCC algorithms get different performance in their prediction. From the result, we can see that the Minimum Covariance Determinant get the best performance which can be 88.68% correctness rate and test time can be 104.55 seconds. At the same time, the oneclassSVM get the lowest performance. After optimization, the correctness rate is only 71.82% when testing the testing data. And it take about 136.14 seconds in the testing part. Isolation Forest also get a good performance is testing, which is 84.32% correctness rate and take 147.38 seconds testing time. In our testing part, we pay much time on testing, and small portion on prediction. So the real time for predictions is less than we get from the result. Here are some result we get from our terminal.

For isolation tree, left is the no optimized one, right one has optimized. We can see that we save a lot of prediction time after optimization, from 559.33 seconds to 147.39 seconds. And also we improve the correctness rate with optimization. Its performance relies heavily on its parameters.

```
319.02837920188904 2579 423 0.8590939373750833
374.5292181968689 5167 835 0.8608797067644118
429.888384103775 7643 1359 0.8490335481004221
485.28855299949646 9941 2061 0.8282786202299617
540.7818171997997 12215 2787 0.8142247700306626
For Test Set: err: 3044 right: 12958 rate 0.8097737782777152 Time: 559.3307881355286
```

```
76.86326503753662 2808 194 0.9353764157228515
93.08142423629761 5594 408 0.9320226591136288
109.470379114151 8204 798 0.911353032659409
125.73615002632141 10489 1513 0.8739376770538244
141.96792912483215 12752 2250 0.8500199973336888
For Test Set: err: 2508 right: 13494 rate 0.8432695913010874 Time: 147.38618636131287 same err:
5884 err_total 5070
```

For Minimum Covariance Determinant, we use EllipticEnvelope to build model. And also left is the no optimized one, right one has optimized. We have improved the correctness from to 88.68%. When we set the suitable value of contamination

```
51.73915910720825 2970 32 0.9893404397068621
63.60298228263855 5949 53 0.9911696101299566
75.57449698448181 8664 338 0.9624527882692735
87.57471013069153 10848 1154 0.9038493584402599
99.69570517539978 13031 1971 0.8686175176643114
For Test Set: err: 2256 right: 13746 rate 0.8590176227971503 Time: 103.82582092285156
```

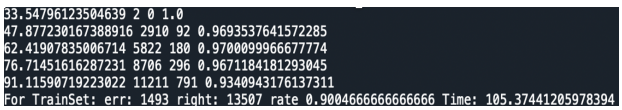
```
53.26978874206543 2910 92 0.9693537641572285
65.08912396430969 5823 179 0.9701766077974009
76.90770792961121 8562 440 0.9511219728949123
88.72285795211792 10983 1019 0.9150974837527078
100.54379796981812 13405 1597 0.893547269964005
For Test Set: err: 1811 right: 14191 rate 0.8868266466691663 Time: 104.54624676704407 same err: 30
err_total 1781
```

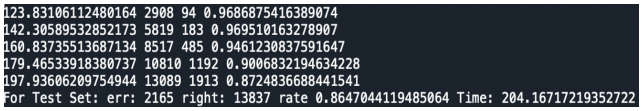
For one class SVM, we have tried a lot to improve the performance. But we get a little performance in return. Also, left is the no optimized one, right one has optimized. This model's best correctness is 71.82%. It is not suitable for detecting DNS based C2 channel.

```
74.89019298553467 2582 420 0.860093271152565
87.51851296424866 5166 836 0.8607130956347884
100.15393710136414 7432 1570 0.8255943123750278
112.80788397789001 8979 3023 0.7401253124479254
125.44338703155518 10565 4437 0.7042394347420344
For Test Set: err: 4924 right: 11078 rate 0.6922884639420073 Time: 129.71910905838013 same err:
4196 err_total 5363
```

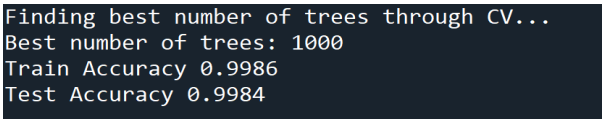
```
80.08344101905823 2438 564 0.8121252498334444
93.06126379966736 4863 1139 0.8102299233588803
105.98448395729065 7123 1879 0.7912686069762275
118.90323781967163 8987 3015 0.7487918680219964
131.83040308952332 10871 4131 0.7246367151046527
For Test Set: err: 4509 right: 11493 rate 0.71822272215973 Time: 136.13885116577148 same err: 117
err_total 4392
```

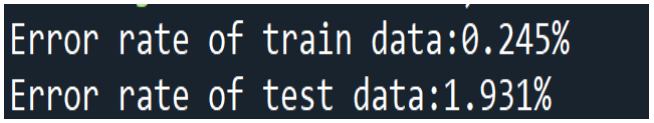
AdaBoosting with average weight: the result get the value from all three OCCs with average weight, we can get the correctness in train data set (left) and test data (right) set below.





AdaBoosting accuracy: The accuracy is about 99% and the sklearn version can also achieve 98% accuracy.





1.5 **Challenges**

1) We only focus on FQDN regardless of any other information, which causes the input of our model much smaller than others, likely leading to a worse result.
2) Hyper feature generation from FQDN.
3) Use AdaBoosting and other methods to combine our OCC models into a stronger one.
4) Capture and parse DNS packets; Using Bloom-filter to build a FQDN whitelist.
5) Evaluate our program in practice and analyse the results on different DNS C2 tools.

2 **BACKGROUND**

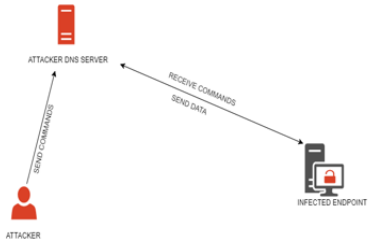
The C2 Channel is a management and control infrastructure, equipment and technology. Attackers use it to maintain contact with infected devices after initial use. The specific method varies from attack to attack, but C2 generally consists of one or more hidden communication channels between a device organized by the victim and a place controlled by the attacker. These communication channels are used to send instructions to the infected device, transfer other malicious loads and recover stolen data from the opponent through the channel.

DNS-based C2 channel has gained its popularity a lot In this years. Many DNS tunneling Tools like dns2tcp, dnscapy, iodine, tuns has developed a lot recently. But also, DNS tunneling is really peculiar among all the C2 channels, which makes it much harder to be detected and blocked. So it is necessary to find a efficient way to detecting DNS-based C2 channel.

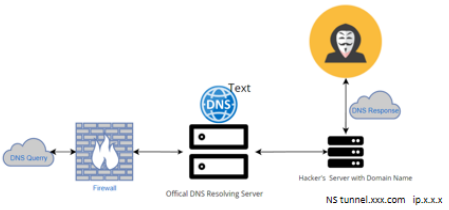
One-Class Classification Algorithm is a unsupervised learning algorithm that attempt to model “normal” examples in order to classify new examples as either normal or abnormal. It involves fitting a model on the “normal” data and predicting whether new data is normal or an outlier/anomaly. Algorithms aim at capturing characteristics of training instances, in order to be able to distinguish between them and potential outliers to appear. Including SVM, isolation forest, elliptic envelope, and local outlier factor and so on. It can be effective for imbalanced classification datasets where there are none or very few examples of the minority class, or datasets where there is no coherent structure to separate the classes that could be learned by a supervised algorithm. For our project, there are only two types of channel here, which can be seen as 0 and 1, so it is suitable to use OCC for detecting DNS-based C2 channel.

3 DISCUSSION

Through different kinds of methods of OCC and subsequent optimization, the detection accuracy is relatively excellent. However, compatibility may not be particularly strong in the types of attacks identified. Because the data is self-generated, some new problems may arise when dealing with more extensive and complex data.

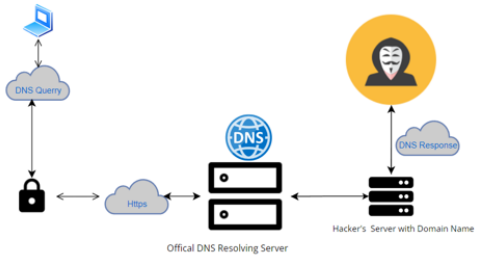


type 1) Direct Connection



type 2) Resolving Domain Name

Also, we point out that DNS C2 can be divided into 3 types. The first type ,we call it “Direct Connection” type, which stores encrypted information in the TXT section of DNS packets and directly communicates from Jump Server to the Victim's Client. Also, type 1 is really similar to those C2 on other protocols and easy to be detected by traditional IDS. The second type is the “Resolving Domain Name” type which we are focusing on. The third type is “DNS over Https”, which is no way to be decrypted, but easy to be banned.



type 3) DoH

```
DNS with qname: social-overlay-launchdarkly-api-prod.ol.epicgames.com sent at time: 01:07:28.227828
Test Result: Malicious
```

fig: False Positive examples for epic-game

For the whitelist, we regard it as a double-edged sword. On the one hand, it will decrease the “False Positive” rate harshly. On the other hand, it may apply extra risks that hackers controlling the websites on the whitelist will easily use DNS C2 without screening.

4 CONCLUSION

We implement One-Class SVM, Minimum Covariance Determinant, Isolation Forest three OCCs for detecting DNS-based C2 channel and optimize the model towards our data set. We get the best correctness rate 88.68% with the Minimum Covariance Determinant model and the best testing time 104.55 seconds. Also we have implement the adaboosting, the average weight version get about 86.4% correctness while the normal version get 98.1% accuracy. And finally, we implement the Real-time Detection with whiteList data set which can detect the DNS-based C2 channel in real time.

5 BIBLIOGRAPHY

[1] M. Aiello, M. Mongelli and G. Papaleo, "Basic classifiers for DNS tunneling detection," 2013 IEEE Symposium on Computers and Communications (ISCC), 2013, pp. 000880-000885, doi: 10.1109/ISCC.2013.6755060.

[2] S. Sayadi, T. Abbes and A. Bouhoula, "Detection of Covert Channels Over ICMP Protocol," 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), 2017, pp. 1247-1252, doi: 10.1109/AICCSA.2017.60.

[3] Bezerra, Vitor Hugo et al. “IoTDS: A One-Class Classification Approach to Detect Botnets in Internet of Things Devices.” *Sensors (Basel, Switzerland)* vol. 19,14 3188. 19 Jul. 2019, doi:10.3390/s19143188

[4] M. Aiello,M. Mongelli,G. Papaleo “DNS tunneling detection through statistical fingerprints of protocol messages and machine learning” 28 July 2014, doi:10.1002/dac.2836Citations

DNS C2 Dataset: https://data.mendeley.com/datasets/mzn9hvdcxg/2
Whitelist Domain Name Dataset: http://s3-us-west-1.amazonaws.com/umbrella-static/index.html