Tools for Active and Passive Network Side-Channel Detection for Web Applications

Michael Lescisin  
University of Ontario Institute of Technology (UOIT)  
michael.lescisin@uoit.net

Qusay H. Mahmoud  
University of Ontario Institute of Technology (UOIT)  
qusay.mahmoud@uoit.ca

August 14th 2018  
Baltimore, MD, USA
Introduction

Underlying Concepts

Design

Implementation

Evaluation

Defences

Conclusion and Future Work

---

Agenda

1. Introduction
2. Underlying Concepts
3. Design
4. Implementation
5. Evaluation
6. Defences
7. Conclusion and Future Work
Introduction

- Cryptography as go-to solution for communication security in insecure environments.
- Even under properly implemented cryptosystems, information such as packet timings and session lengths still remains intact.
- Related research has shown that it is possible to learn sensitive information from observing these information features.
- Unlike more direct information leaks such as SQL Injections, the current state of research for detection tools based on known-to-be-vulnerable design patterns still lags behind.
SSL/TLS

- Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS) are popular cryptographic protocols for maintaining the confidentiality, authenticity and integrity of a web session.
  - Works by using public key cryptography to exchange a symmetric encryption and authentication key.
  - A list of certificate authorities is trusted to endorse, by public key signature, the public key certificate for an SSL/TLS using website.
  - The protocol is used on every HTTPS URL.
The protocol is designed to maintain *confidentiality, authenticity, and integrity* even if *network traffic can be intercepted or manipulated*.

The protocol *should* allow a developer to wrap their plaintext TCP connections with SSL/TLS and the above three information security properties should be upheld.

Any violation of these properties would be a case where SSL/TLS does not perform sufficiently to satisfy its threat model.
Practically Speaking

- Wi-Fi snooping
- Compromising internal network equipment for ARP/IP/DNS spoofing
- Malicious VPNs or proxies
- Malicious ISPs
- Illegal wiretapping
- Etc...
A fundamental understanding of the interaction between computer networks and modern web development is necessary for understanding the side-channel vulnerabilities in SSL/TLS.

One must be aware that SSL/TLS does not hide the following information:

- Approximate size of data transferred in a session
- Start and end times of each session
- IP addresses and domain names of clients and servers
- Order of sessions
Let us consider measuring the approximate size of a session:

- Maximum payload size of Ethernet frame is **1500 bytes**.
- Minimal size of IP header is **20 bytes**.
- Minimal size of TCP header is **20 bytes**.
- Therefore maximum HTTP payload carried in one Ethernet frame is **1460 bytes**.
- Our empirical observations have shown the maximum HTTP payload size to be **1370 bytes**.

Therefore session size can be estimated by finding the sum of continuous sequences of **1370 bytes**, plus the size immediately before, plus the size immediately after.
Network Traffic Features

**Figure**: An approximation of session length can be recovered by summing continuous sequences of 1370 bytes.
Network Traffic Features

- Other useful features within a network traffic sample include:
  - Bursts of network activity
  - Timing between activity bursts
  - DNS information
  - Packet counting
Modern Web Development

- Web *applications*, not just web *pages*.
  - Real-time client-server communication (AJAX).
  - Responsivity
    - Lazy-loading of resources
    - Scalable protocols (ie. DASH)
Be able to exploit the following three *modern* web designs:

- Response Dependant Page Loads
- Real-time Feedback Systems
- Lazy Loading
Exploiting Response Dependant Page Loads

- Remember, SSL/TLS is supposed to protect the confidentiality of a web application.
- Therefore, an adversary observing the encrypted communications of a web browser should be able to learn nothing on what the user has entered into an HTTPS submitted form.
- But what if the next page to be loaded depends on the responses submitted through this form?
Remember, using SSL/TLS does not hide the sizes of sessions nor the times at which they begin and end.

Therefore, an adversary can learn:
- When the real-time event occurred
- The possible type(s) of the real-time event based on its size
A webpage with lazy loaded images will make different network requests dependant upon the geometry of the viewport.

Remember that an adversary can *open and close* popup windows *in arbitrary sizes* to *cross-domain* webpages!
Exploiting Lazy Loading

Therefore an adversary can learn about the page layout by observing the network traffic generated when loaded in different sizes.

```javascript
function do_popup(){
    var TARGET_URL = "https://targeturl.com/targetpage";
    popup_window = window.open(TARGET_URL,'targetWindow','toolbar=no,location=no,status=no,
    menubar=no, scrollbars=no, resizable=no, width=100, height=200');
    //Set a timer to close automatically close this window
    window.setTimeout(function(){
        popup_window.close();
    }, 10000);
}
```
Our tools were built with the help of several open-source software packages.
This section discusses those packages which play primary roles in our tools.
Understanding these packages is also necessary for future improvements to our tools.
We would like to make our tests:

- Isolated from normal computer use
- Easily distributable and reproducible

Docker is the perfect tool for this!

Dockerfile

```bash
FROM ubuntu:latest
RUN apt-get update
RUN apt-get install -y vnc4server jwm firefox
RUN apt-get install -y tcpdump
...```

Michael Lescisin, Qusay H. Mahmoud
University of Ontario Institute of Technology (UOIT)

Tools for Active and Passive Network Side-Channel Detection for Web Applications
A subsystem of the Linux kernel allowing for network packet filtering and manipulation.

- Most commonly used through the `iptables` command.
  - **Netfilter Queue (NFQUEUE)** target of `iptables` allows for network traffic to be manipulated through user space programs!

```python
def process(i, payload):
    payload.set_verdict(nfqueue.NF_DROP) # Drop all packets

q = nfqueue.queue()
q.open()
q.bind(socket.AF_INET)
q.set_callback(process)
q.create_queue(0)
```
Scapy

- A very robust Python library for packet creation, manipulation, and inspection.

```python
from scapy.all import *
cap = rdpcap('captured_example.pcap')
len(cap)
>>> 1889
cap[281][IP].dst
>>> '172.19.0.2'
cap[281][IP].src
>>> '52.84.143.116'
cap[281][IP][TCP].sport
>>> 443
cap[281][IP][TCP].dport
>>> 44782
len(cap[281][IP][TCP].payload)
>>> 1370
```
We have discovered that through monitoring network traffic an adversary can:

- Discover which political candidate is recommended for a user of *iSideWith.com*.
- Determine which of the top ten Google searches of 2017 a user is likely searching for.
- Estimate the number of items in a user’s eBay shopping cart.

These examples demonstrate the exploitation of the three vulnerable web designs discussed in this presentation; response dependant page loads, real-time feedback systems, and lazy-loading of web resources.
iSideWith.com recommends a political candidate based on a user’s responses to questions on divisive political issues.

After submitting the questionnaire, a page displaying the recommended candidate is loaded.

Therefore, the web design to be exploited in this example is \textit{Response Dependant Page Loads}.

Michael Lescisin, Qusay H. Mahmoud

University of Ontario Institute of Technology (UOIT)

Tools for Active and Passive Network Side-Channel Detection for Web Applications
Introduction

Underlying Concepts

Design

Implementation

Evaluation

Defences

Conclusion and Future Work

Tools for Active and Passive Network Side-Channel Detection for Web Applications

Michael Lescisin, Qusay H. Mahmoud

University of Ontario Institute of Technology (UOIT)
iSideWith.com

- Collected 80 PCAP network traffic capture files.
  - 40 corresponded to a recommendation of Donald Trump.
  - 40 corresponded to a recommendation of Hillary Clinton.
  - Captured data was split 50/50, training/testing.
- Evaluated various machine learning classifiers.

![Evaluation of Trained Classifiers](image)

- True Positive Rate (Clinton), True Positive Rate (Trump)

Michael Lescisin, Qusay H. Mahmoud
University of Ontario Institute of Technology (UOIT)
Tools for Active and Passive Network Side-Channel Detection for Web Applications
Analyzing Google Auto-Suggest

Every keystroke entered into the Google Search bar results in a burst of network traffic sent to the server containing the search substring followed by a burst of network traffic sent from server to client containing the suggested search queries.

By observing this encrypted network traffic, an adversary can learn:
- The approximate number of keystrokes entered.
- The sizes of the suggestion lists.
Figure: By observing the timing of network packets, counting the number of keystrokes entered becomes a trivial task.
For analyzing the Google Auto-Suggest traffic, the following machine learning approaches were taken:

- Bayesian classification using number of bytes sent from server to client in each burst of network activity as well as the total number of bursts, as features.
- Nearest neighbour classification using the total number of exchanged packets in a session.
- A *hybrid* approach where the score is calculated as the Bayesian score divided by one plus the packet count distance.
Analyzing Google Auto-Suggest

Evaluation of Bayesian Classifier
(True Positive Rate, False Positive Rate, Non-Detection Rate)

Evaluation of Hybrid Classifier
(True Positive Rate, False Positive Rate, Non-Detection Rate)

Evaluation of Nearest Neighbor Classifier
(True Positive Rate, False Positive Rate, Non-Detection Rate)

<table>
<thead>
<tr>
<th>Number</th>
<th>Search Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mayweather vs McGregor Fight</td>
</tr>
<tr>
<td>2</td>
<td>Las Vegas shooting</td>
</tr>
<tr>
<td>3</td>
<td>Hurricane Harvey</td>
</tr>
<tr>
<td>4</td>
<td>Solar Eclipse</td>
</tr>
<tr>
<td>5</td>
<td>Matt Lauer</td>
</tr>
<tr>
<td>6</td>
<td>Fidget spinner</td>
</tr>
<tr>
<td>7</td>
<td>Aaron Hernandez</td>
</tr>
<tr>
<td>8</td>
<td>Tom Petty</td>
</tr>
<tr>
<td>9</td>
<td>Hurricane Irma</td>
</tr>
<tr>
<td>10</td>
<td>Super Bowl</td>
</tr>
</tbody>
</table>
Remember that SSL/TLS *should* protect the *integrity* of a web session.

But if certain functionality can be detected and blocked then the *integrity* property is violated.

What if we feed live network traffic to the Google Auto-Suggest classifier and use the output to control network traffic policy?
drop_traffic = False  # Should traffic be dropped or not
captured_session = []
def process(i, payload):
    global drop_traffic
    data = payload.get_data()
    if drop_traffic:
        print "!!! Your Internet has been censored !!!"
        ...
        payload.set_verdict(nfqueue.NF_DROP)
    return
    ...
    # Try to classify this session
    hybrid_result = getHybridClassifierType(captured_session,...)
    print "!!! Hybrid Result is {}. Length of session is {} !!!".format(...)
    for censor_item in CENSORSHIP:
        ...
        if hybrid_result == censor_label:
            if len(captured_session) in range(size_min, size_max):
                drop_traffic = True  # Flip the killswitch!
Censoring Google Auto-Suggest
Counting eBay Cart Items

- Remember how on a page with lazy-loaded elements only the elements which are within the viewport are downloaded over the network.
- Therefore, if expanding the geometry of the viewport causes additional network traffic, this implies that lazy-loaded elements are present.
- eBay’s shopping cart uses lazy-loading. Therefore we can exploit this web design to estimate the number of items in the shopping cart.
  - We can read DNS replies and isolate traffic from eBay’s image content delivery network (CDN).
Counting eBay Cart Items

**Popup Side-Channel Example**

![Popup Side-Channel Example](image)

**CDN Accesses vs. Shopping Cart Item Count**

Michael Lescisin, Qusay H. Mahmoud

Tools for Active and Passive Network Side-Channel Detection for Web Applications
Defences

To prevent against these types of attacks two conditions must be satisfied:
- Network traffic patterns must be *indistinguishable* from each other.
- The user and web application should immediately be made aware of network traffic tampering (e.g., blocking requests).
Consider the implications of strict adherence to the condition that *network traffic patterns must be indistinguishable from each other*:

- All web requests must be padded to match the size of the largest web request.
- *Noise* traffic must be generated to obscure the presence of real-time events.

Not suitable as a general-purpose fix as this would result in intolerable performance overheads for many applications.
Defences

- At the current state of research an *application-specific threat model* and a *cost-benefit* approach are required.
  - Does revealing what page was loaded violate the security requirements?
  - Does revealing the timing of user events violate the security requirements?
  - Etc...

- The model-driven software development process can help.
  - Assign a property that all pages of a class be of the same size.
  - Assign a property that the timing of network events *must* follow a predefined schedule.
  - Etc...
Conclusion

- We have discussed the theory and implementation for side-channel detection tools capable of detecting the following types of network traffic based side-channels in web applications:
  - Response dependant page loads
  - Real-time feedback systems
  - Lazy-loading of web resources
- We have determined these vulnerable designs to be present in modern web applications.
- We have released all source code and example network traffic capture files on our GitHub page.
Future Work

- Work with model-driven software development to enforce side-channel resistance properties such as those discussed in the *defences* section.
- Expand the future model-driven development work to include non-web network services such as remote shells (SSH) or control of IoT devices.
Thank you!

Any questions?

https://github.com/uoitdnalab/networksidechannel