Network Security

Ex. 1

CensorSpoofer

Article Report

By:

Ron Meiry	301224283
Slava Ustinov	309930006

CONTENTS

ARTICLE SUMMARY

INTRODUCTION

In the past decade, the Internet was used extensively by independent users and groups as a platform for protest and organization against oppressive governments. These events have caused such governments, China for example, to enforce a strong censorship policy against web-sites or services that were used extensively by protesters, such as Twitter and Facebook, to share ideas that are contrary to the government's political views.

GOAL

The article proposes a new scheme to allow users access to restricted web-sites, while remaining undetected by an attacker, which is an oppressive government in our case.

ATTACKER MODEL

Our model adversary is of government level. It's goal is to identify the rogue users and to enforce it's censorship policy. The attacker has all plausible capabilities expected from a government:

1. Full control over the network's infrastructure.
2. Ability to perform DPI, DNS hijacking, IP filtering and Packet injection.

Though our adversary is powerful, it's not omnipotent. For example, it's very unlikely that he's able to crack a modern day encryption (such as AES, RSA, etc.).

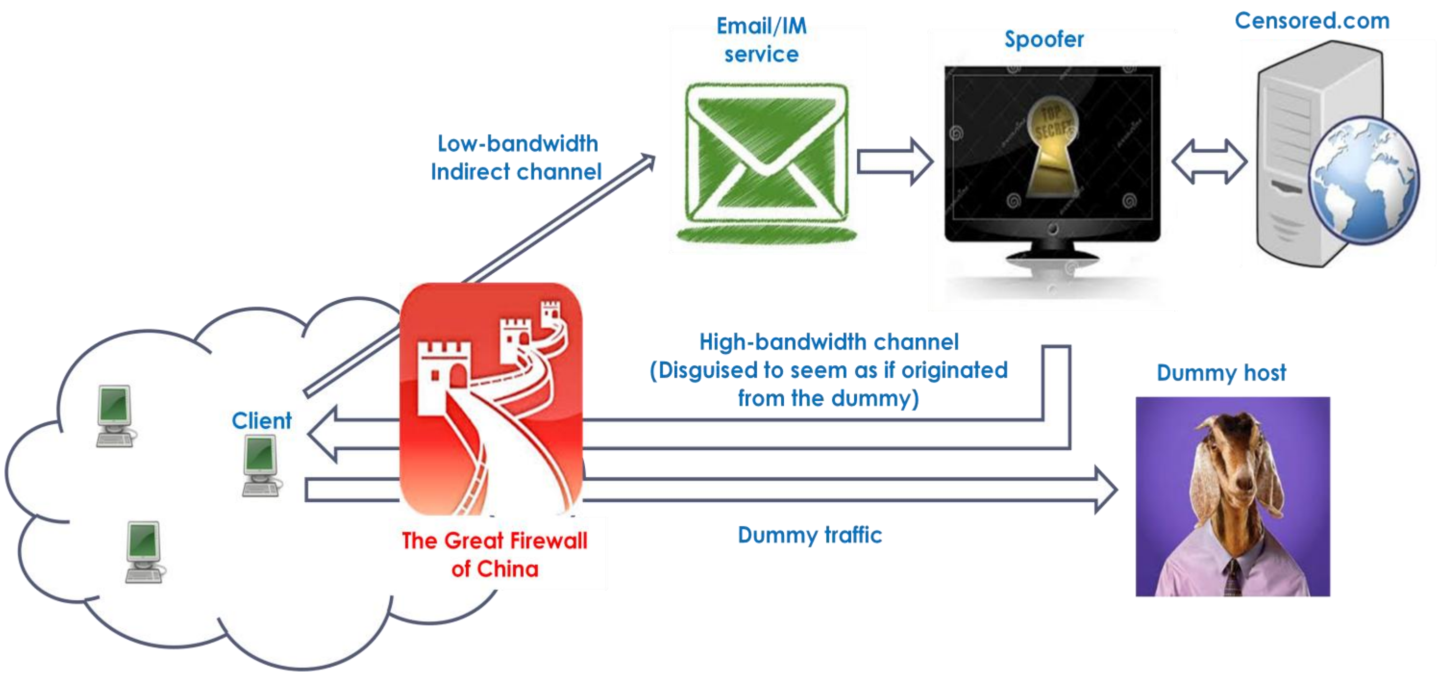
Given the capabilities described above, the attacker is motivated do avoid economic damages that may be caused by extreme actions, such as blocking all e-mail services or preventing usage of mainstream encrypted communication protocols.

PROPOSED SCHEME

ENTITIES

The article's proposed scheme has 3 main entities:

- Client - Resides within the domain of a censoring authority.
    - Desires to gain access to a restricted web page, while remaining undetected.
- Spoofer server - Resides in a domain without any censorship policy.
    - Acts as a web proxy, according to the client's requests.
    - Wishes to remain untraceable and unobservable by any censoring authority, therefore it has to restrict it's identifiers to the client.
    - Wishes to be invulnerable to insider attacks (attacker agents disguised as legitimate users).
- Dummy host - An unaware machine that resides in a domain without any censorship policy, that's used as a scapegoat. The Dummy's IP address is used by the Spoofer to create a seemingly legitimate traffic.



Email/IM service
Spoofer
Censored.com
Low-bandwidth Indirect channel
High-bandwidth channel (Disguised to seem as if originated from the dummy)
Dummy host
Client
The Great Firewall of China
Dummy traffic

REQUIRED SETUP:

1. Both the Spoofer and the client are using SIP (5th layer protocol) to support VoIP communication over UDP.
2. Since SIP required SIP ID (for both target and source) to establish communication, the client must have a valid SIP ID that is used by the Spoofer.
3. The client must be familiar with an email/IM account that is used by the Spoofer.
4. Steps 2-3 can be avoided if the new user is trusted by an already existing user:
    a. The new user creates a new SIP ID and a new email/IM account.
    b. The new user passes these details to the Spoofer via an existing user.
5. The communication is encrypted.

WORKFLOW:

1. The client initializes a SIP session with the Spoofer:
    a. The client sends out an INVITE message, which is targeted to the Spoofer's SIP ID, towards well known SIP servers.
    b. The SIP servers forward the INVITE message towards the target.
    c. Once the message has reached the target, the target can send an OK response.
2. The Spoofer, which obviously cannot give away it's actual IP address, responds with a manipulated OK message:
    a. It first picks a dummy host which actually supports SIP.
    b. The IP address in the OK message is spoofed with the dummy's IP address, to seem like it originated from the dummy.
3. The client starts sending encrypted traffic with random content towards the dummy. The traffic matches the pattern of VoIP sessions. The dummy simply discards this traffic.
4. The client can now send URL requests to the Spoofer via the indirect channel.
5. The Spoofer receives the client's request, fetches the requested web pages and sends them towards the client, while disguising the traffic as legitimate VoIP traffic. The source IP of these packets is spoofed to be the dummy's IP address.

CRITICISM

THE USAGE OF UDP

The article suggests usage of UDP as the 4th layer protocol. UDP was chosen because we can't use TCP for an asymmetric communication channel, as suggested by the article.
The proposed scheme suggests masking HTTP traffic as VoIP traffic, meaning the packets can't tolerate big delays and therefore must be relatively small and frequent. This forces a single web page to be broken down to multiple small packets.
Since UDP has no retransmission mechanism, lost packets will never be retransmitted. The article suggests implementation of FEC (Forward Error Correction) codes, which can tolerate a loss of a certain amount of packets.
What'll happen in a case where an intolerable amount of packets is lost?

This scenario can be exploited by the attacker:
Since the nature of the communication between the client and the Spoofer is asymmetric, retransmission requests can only be sent over the indirect channel.
The attacker can monitor the usage of email/IM services by a suspected client, in correlation with the amount of dropped VoIP packets (which can be controlled by the attacker). The ratio between lost VoIP packets and the amount of generated email/IM packets by a user can indicate the usage of a Spoofer.

Another viable point is the network congestion that'll occur in such scenario:
The loss of UDP packets will cause an increasing rate of retransmission requests via email/IM, adding load on an already overloaded network.

VULNERABILITY TO DOS ATTACKS

The asymmetric nature of the communication between a client and the Spoofer leaves the Spoofer exceptionally vulnerable to DoS attacks, especially DDoS.
Let's assume the following scenario:
The attacker creates multiple clients and registers them with the Spoofer, which is completely plausible with our attacker's model.
From this point on, these malicious clients can simultaneously bombard the server with requests. This attack will be extremely efficient if the requested web pages will be as large as possible.
While a relatively small effort is required from each malicious client to launch such attack (simply sending a short email), the Spoofer's operation is much more complex:
It has to fetch large web-pages, break them down to small packets, encrypt and wrap this data with the valid protocol headers.
This'll cause a computational load (encrypting lots of messages) and a massive consumption of bandwidth (lots of headers for small blocks of data).

The attacker can also manipulate innocent users to aid it's cause by simply choosing a popular, 'heavy' website and simply blocking it. This'll create a surge of requests for that website by regular users, adding to the load on the Spoofer.

Alternatively, the attacker can create a valid, 'heavy', desirable website and use it as bait for a short time period (to gain popularity among users) and then to block it, thus causing the scenario described in the paragraph above.

RESEARCH QUESTIONS

QUESTION #1 - 'FINGERPRINT' ATTACK

BACKGROUND:

The suggested scheme in the article masks HTTP traffic as VoIP.
We suspect that the fingerprint (delay variance, packet size variance, etc.) of normal VoIP traffic might be different from the traffic masked as VoIP that the scheme presents.
Our suspicion rises from the following reasons:

1. The client generates pseudo-random VoIP packets.
2. The Spoofer's packets contain data from a textual source (HTML) and not actual voice data.

Another unique behavior of the protocol is the usage of an email/IM service shortly after the initialization of a VoIP session.

SUGGESTED ATTACK:

We've already presented the possibility of malicious clients, organized by the attacker, using the Spoofer legitimately.
Machine learning might be implemented by the attacker:
Such users can be used for statistical information gathering on the fingerprint of such traffic (which is disguised as VoIP). The attacker can build a large data set, calibrate key classifiers associated with the traffic and create a fingerprint. From this point on, the classifiers will continue to be updated in real time, to prevented false indications that may be caused by topology changes and traffic variation.
A similar analysis can be made on a genuine VoIP session.

The attacker can create 2 indicators:
Indicator #1: Does a specific VoIP session match the normal VoIP pattern?
Indicator #2: Does a specific VoIP session match the Spoofer's protocol?

The 2nd indicator is a non-trivial and a powerful attribute, which allows us to detect the specific anomaly that we're interested in.

From now on, the attacker can use the classifiers above to find anomalies in ongoing VoIP sessions.

SUGGESTED MITIGATION

Both the client and the Spoofer can use the methods above and derive the fingerprint for a normal VoIP session.
When they've done so, they must mask their traffic to match the normal distribution of genuine VoIP traffic. Furthermore, the client should reduce the correlation between the timing of VoIP session initiation and the sending of information on a side channel. It can do so by adding random delay times and choosing from multiple side channels, as esoteric as possible (possibly messaged via various chat rooms).

QUESTION #2 - TRAFFIC MANIPULATION ATTACK

BACKGROUND:

The scheme suggests the following operation:
The client sends encrypted VoIP packets, with some random content toward the dummy. This traffic will surely be discarded by the dummy.
VoIP devices, using RTP/RTCP (5th layer control protocols for VoIP) as suggested by the article, can tolerate up to about 30 seconds of complete packet loss. In other words, after 30 seconds of no life signs, the protocol terminates the connection.

ATTACK SUGGESTED IN THE ARTICLE:

The article suggests discovering the use of a Spoofer in the following way:
Dropping all VoIP packets to/from a suspected user for 30 seconds and monitoring the traffic. A normal VoIP session should be terminated, while the usage of a Spoofer won’t be, VoIP packet will continue flowing in both directions.
The writers classify this attack as infeasible since the attacker can't tell which VoIP packets belong to a certain flow, so he has to drop all VoIP packets associated with a specific user. Since the attacker has no feasible way to identify suspected users, this process is highly inefficient because it has to be performed on all VoIP-using users.

SUGGESTED IMPROVEMENT:

We suggest improving the attack described above in the following way:

1. We'd narrow the entire user space to a smaller space of users suspected in user a Spoofer. We suggest doing so by using the 'Fingerprint Attack' described above.

2. For each suspected user, we'll perform the following:
    a. For 30 seconds, we'll drop all outgoing VoIP packets.
    b. If incoming VoIP packets will continue to arrive steadily, we'll have a good indication of a Spoofer usage.

We believe the suggested 'surgical' method will reduce the attacker's required resources and will yield more accurate results.

QUESTION #3 - DISTRIBUTED NETWORK OF SPOOFERS

BACKGROUND:

The article suggests usage of a single Spoofer server to provide the described anti-censorship ability. As we've discussed before, this leaves the system susceptible to DoS attacks, since there's a single point of failure.

SUGGESTED ENHANCEMENT:

We suggest deploying a cloud-like network of available Spoofers.
Our main goal is to improve the robustness of the CensorSpoofer functionality, while remaining true to all other features of the article's scheme.

ELABORATION:

Suggested feature: Load balancing
Implementation: We suggest distributing the responsibility for segments of side-channel IDs between a group of servers; meaning, every server will be exclusively responsible for a set of side-channel IDs (i.e. email accounts).
Issue:

> This implementation doesn't DoS attacks on a specific server, since every side-channel ID is associated with single server only.

Solution:

> To mitigate such attack, we suggest implementing a proxy mechanism between the Spoofer, that'll allow an overloaded Spoofer to use another Spoofer as a proxy for the request. Since the Spoofer may reside in different physical areas, we can use IP tunneling to implement the mechanism described above.

Suggested feature: Survivability

Implementation: We suggest creating a logical full-mesh between the Spoofers.

Issue:

What'll happen if one of the Spoofers crashes?

An entire set of side-channel IDs will become unavailable.

Solution:

We suggest the following protocol:

1. All Spoofers have the entire side-channel IDs data set.
2. One of the Spoofers will be chosen as leader (using a distributed alg.).
3. The leader will keep track of the state of the other Spoofers.
4. If a peer Spoofer crashes, the leader will distribute it's responsibilities between all other active Spoofers.

Suggested feature: Global, Distributed Web Cache mechanism

Implementation: We suggest keeping a small web cache on each Spoofer, that'll hold the most popular web pages. Once in a while, the Spoofers will share their local caches among themselves (URLs are sufficient), either through the full-mesh or through a leader (who'll be responsible for pulling the information and pushing it to all peer Spoofers).