Analytical and Practical Evaluation of Sybil Attacks

Andrew Rosen

November 20, 2014

Abstract

This paper explores the feasibility of performing naive Sybil attacks that completely occlude healthy nodes from each other. The vulnerability of Distributed Hash Tables to Sybil attacks and Eclipse attacks has been well known for some time. However, these vulnerabilities has often been explored in a theoretical sense, assuming the attacker is a global adversary from the beginning, nigh-omniscient and omnipotent. This paper seeks from a analytical and practical perspective, how valid that assumption is.

We examine the amount of computational effort required to become a global adversary starting as a non-global adversary. We do this by analyzing the amount of time it takes an attacker with a given IP to choose a port to obtain a desired hashkey, a process we call *potatoing*. We present potatoing to emphasize the ease of this attack, but also demonstrate potential non-security uses of potatoing that are beneficial to DHT load-balancing.

1 Introduction

One of the key properties of structured peer-to-peer (P2P) systems is the lack of a centralized coordinator. P2P systems remove the vulnerability of a single point of failure, but in doing so, open themselves up to new attacks. Most P2P systems and distributed hash tables (DHTs) are vulnerable to *Eclipse attacks*, whereby an attacker completely occlude healthy nodes from one another. This prevents them from communicating without being intercepted by the attacker.

One way to accomplish this attack is to perform a `Sybil Attack`. In a Sybil attack

Security analyses typically assumes an omnicient attacker.

I want to practically demonstrate this as well demonstrating how easy it is to place nodes in regions.

A sybil attack is named after Sybil, who had Dissociative Identity disorder. In a sybil attack, the attacker

This is a lot like a birthday attack, only searching for a collision with a region

Why am I doing this? DHTS are important cause I like them.

Security is not something that is thought about for a DHT, unless the DHT is specifically made to be secure against X. Or it's left to the applications

A complete DHT occlusion is overkill.

- We first discuss the mechanics of performing a Sybil attack and analyze it's effectiveness from a analytical standpoint.
- Our experiments demonstrate the trivial amount of time and computational effort needed to

2 Analysis

[?]

The birthday attack analysis says given so many elements, likelyhood of collision between any of these elements. The potato attack says given this region, what is likelyhood i can find something in this region. It's a different analysis since I'm looking for 1 attacker colliding with one specified region at a time.

Suppose we have a DHT with N members in it, with the hashspace of $[0, 2^{160})$. The case of small N is ignored, since they are trivial even when unbalanced. We can assume that, for a large enough N, node IDs will beclose to evenly distributed accross the network, meaning there will be $\approx \frac{2^{160}}{N}$ hashkeys between each node ID.

Size of bin is

$$bin = \frac{H}{N}$$

The probability P of an attacker finding an hashkey that lands in the range with the range (n, m) is

$$P \approx \frac{n-m}{H} \cdot num\_ips \cdot num\_ports$$

making it equivalent to

$$P \approx \frac{1}{N} \cdot num\_ips \cdot num\_ports$$

The chances of compromising an entire network of N nodes that partition the entire network.

$$c \approx 1 - (1 - \frac{1}{N})^{P*I})$$

Alternatively, we can view this as doing a birthday attack in progress with different probabilities. EG, we've generated $\frac{h}{N}$ values already, how many more do we need?

3 Simulations

Simulations were performed on a computer with consumer-grade budget hardware.

3.1 Experiment 1:Potatoing 2 random nodes

Our initial experiment was designed to establish the feasability of injecting in between two random nodes.

Each trial, we generated two victims with random IP/port combinations, and an attacker with a random IP. The experiment was for the attacker to find a hashkey in between the two victim's key, from the lowest to highest.

The amount of time to potato two random hashkeys was 29.6218323708 microseconds, and was achevied 99.996% of the time.

3.2 Experiment 2: Nearest Neighbor Eclipse via Sybil

The objective of the second experiment is to completely ensnare a network using a Sybil attack, starting with single node.

3.3 Experiment 3: Fully Complete Eclipse via Sybil

The previous experiments of

In some of the below frameworks, it is more efficient to perform an Eclipse attack by falsely advertising, rather than injecting

3.3.1 Chord

Most of the above attack

Chances the finger is already covered:

3.3.2 Kademlia

3.3.3 Plaxton Based networks

More efficient to lie.

4 Masking the Attack

Now that we have established that a Sybil attack can be performed with great ease, our focus now turns to avoiding detection.

We need a different IP for each point surrounding our victim. In the Nearest-Neighbor attack, we need a

We can reduce this into an interesting graph coloring problem

5 Simple Load Balancing Injection Framework

Costs: Need to hold 15000ish hashkeys, effectily 160-bit numbers

References