Analytical and Practical Evaluation of Sybil Attacks

Andrew Rosen

November 19, 2014

Abstract

This paper explores the feasiblity 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-omnicient 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 benificial to DHT load-balancing.

1 Introductions

Security analysis typically assumes an omnicient attacker.

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

Most DHTs are vulnerable to a *sybil attack* or *eclipse attack*.

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.

2 Analysis

[1]

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[1] 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.

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 demonstrate the 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: Naive Complete Eclipse via Sybil

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

References

[1] Mihir Bellare and Tadayoshi Kohno. Hash function balance and its impact on birthday attacks. In *Advances in Cryptology-Eurocrypt 2004*, pages 401–418. Springer, 2004.

[1] p