Introduction

Services on the public Internet are frequently scanned, then subject to brute-force and denial-of-service attacks. We would like to run such services stealthily, available to friends but hidden from adversaries. In this work, we propose a moving target defense named “Chhoyhopper” that utilizes the vast IPv6 address space to conceal publicly available services. The client and server hop to different IPv6 addresses in a pattern based on a shared, pre-distributed secret and the time of day. By hopping over a /64 prefix, services cannot be found by active scanners, and passively observed information is useless after two minutes. We demonstrate our system with the two important applications—SSH and HTTPS.

This poster presents the design and implementation of Chhoyhopper with SSH and HTTPS applications.

Chhoyhopper Design

Our goal is to allow the client to rendezvous with the server on a public, but temporary IPv6 address. By allocating the temporary address from a large space (2^64 addresses), scanning is impractical.

Address hopping pattern:
- Client and server must follow the same hopping pattern to rendezvous.
- Client and server share a pre-distributed key and salt value.
- The server and the client compute the same temporary address by computing a cryptographic hash (we use SHA-256) of the shared secret, a salt value, and the current time in minutes.
- Using NAT, (i) only the current IPv6 address will be translated to the internal service address, (ii) we keep the existing connections, and (iii) traffic with the other target addresses will be dropped.

Getting rendezvous address:
- We get the IPv6 address of a domain name using DNS.
- We keep the first 64 bits and replace the last 64 bits using the generated value from the hash function with key, salt, and timestamp.
- The server updates the address every minute. To handle clock drift, the server keeps two addresses at a given time.

Risk of Discover and Collisions

Risks of discover and collisions are tiny.

Discovery: not in our lifetime! Scanning at 100 Gb/s, expected time to discover one server in one /64 is 3000 years.

Collision: it takes a million servers to get a collision in 70 years. Collisions are like the birthday problem, but in a “year” with 2^64 days. The probability of collision is $1 - e^{-k^2/2N}$, with $N = 2^{64}$ for k servers.

Conclusions

- We show the design and implementation of Chhoyhopper.
- We plan to provide a Chhoyhopper client as a patch to OpenSSH and provide HTTPS support for Chrome.
- Our implementation for SSH and HTTPS applications is freely available at: https://ant.isi.edu/software/chhoyhopper/.