PA.RECON6

Parasitic Reconnasaince using ICMPv6
SSN Project proposal 20 Sep 2019

Axel Koolhaas
12019364

Paul Dunn
12937479

Philipp Mieden
12786721

Davide Pucci
12809926

September 2019

1 Introduction

The traceroute tool is used identify traversed hosts by incrementing the Time To Live (TTL) field on the IP packet. If an attacker can manipulate a previously established connection through the target network, this principle can be used to identify internal hosts otherwise hidden by firewalls or routers using network address translation. To make this kind of reconnaissance work, either a Man-in-the-Middle (MITM) attack has to be performed to intercept packets and inject probes with a low TTL value. One could also lure the victim to contact an attack controller service e.g. by phishing. Previous security related work focused on IPv4 networks only[1][2], while others used this technique for debugging purposes and not with malicious intent [1][2]. Furthermore the implications of this attack type have not been evaluated for IPv6 networks. Since IPv6 security is maintained separately and the ICMP specification has been modified for the IPv6 protocol, we intend to verify the impact of these changes, both from a technical and operational perspective.

2 Research question

This section presents our main research question and sub-questions that follow from it.

> How effective is parasitic TTL based reconnaissance in discovering paths and nodes in IPv6 networks compared to IPv4 networks while including comparative behavioral analysis and what are practical solutions for mitigating this type of reconnaissance?

[1]`http://www.adeptus-mechanicus.com/codex/paratrc/paratrc.php`
[2]`https://lwn.net/Articles/217076/`

This question can be divided into two sub-questions:

1. Does the tracing yield different paths, nodes, and behavior compared to tracing IPv4-based connections?
2. How can system administrators mitigate and detect this class of attacks?

3 Related work

As already mentioned, other research already deepened on how different traceroute implementations could avoid the common hardening firewall rules by designing new flows and architectures.
The greatest part of related work relies on a TCP-based architecture, which enables probes (that resemble re-transmitted packets) to piggyback on TCP connections, aiming to circumvent the standard firewall traceroute-blocking rules.

3.1 `paratrace` (Parasitic trace)

`paratrace` traces the path between a client and a server, attaching itself to an existing TCP flow, statelessly releasing as many `TCP Keepalive` messages as the software estimates the remote host is hop-distant, proceeding then to analyze the resultant `ICMP Time Exceeded` replies. It was developed by Dan Kaminsky [3].

3.2 `tcptraceroute`

`tcptraceroute` is a traceroute implementation using TCP packets. It differs from the other TCP-based implementations in that it's creating the TCP connection itself on its own and its strength relies on the use of TCP `SYN` packets, which, in most cases, is able to bypass the common firewall filters.
Also, `tcptraceroute` never completely establishes a TCP connection with the destination host. If the target is not listening for incoming connections, it will respond with a TCP `RESET`, indicating that the port is closed; otherwise `tcptraceroute` will get a TCP `SYN/ACK` response back, meaning that the port is known to be open: a TCP `RESET` will be sent by the kernel on which `tcptraceroute` is running, to tear down the connection without completing three-way handshake.

3.3 Paris Traceroute

Paris traceroute is a TCP-based traceroute designed to be able to correctly deal with per-flow load balancers inside the path it's trying to discover by coherently handling the headers of probes it's sending. It's also able to recognize per-packet load balancers but, due to the random nature of such load balancing, cannot

[3]`https://dankaminsky.com/2002/11/18/77/`

perfectly enumerate all paths in all situations. In order to do such things it needs to pair responses to their originating probes:

- For UDP probes, Paris traceroute varies the `Checksum` field, which requires manipulating coherently the payload they're carrying, in order to not get discarded.
- For ICMP `Echo` probes, *Paris traceroute* varies the `Sequence Number` field, as classic traceroute, but also the `Identifier field`, in order to keep the value constant for the `Checksum` field.
- For TCP probes, *Paris traceroute* varies the `Sequence Number` field.

3.4 Service Traceroute

Service traceroute passively listens to application traffic and injects probes that pretend to be part of the application flow, no matter whether it's TCP or UDP. Also, it provides its own calibration system for popular Internet services, with the aim to find the parameters to be used to obtain the best trade-off between the probing network overhead and the amount of information gathered. This is needed as application flows may close before *Service traceroute* ends the tracing, and hence middleboxes along the path may discard its probes.

4 Approach & methods

After analyzing previous work, we will setup an environment where we can locally experiment with ICMPv6 and parasitic reconnaissance. We will most likely build upon previous work, such as tracetrout[4] and the service traceroute implementation[5]. For this research project, we will complete the implementation of IPv6 functionality, as well as extending functionality to suit our needs. When our environment works accordingly and all ethical information discussed in Section 6 has been approved, a notification will be published informing potential volunteers of participating in this experiment. The network operators will most likely be inclined to accept, since it is a free security scan without caveats but potentially providing valuable insights. Nonetheless, we must take into account that we don't illicit a lot of responses. In that case we must base our research on our own premises. This would be disappointing, but not inescapable. After running multiple experiments, we will aggregate and analyze the data and determine how much information we could infer about private hosts in the target networks. We will specifically look for differentiating behavior between IPv4 and IPv6. If time allows us, we will develop a tool for system administrators to evaluate their IPv6 network defenses.

[4]`https://github.com/HowNetWorks/tracetrout`

[5]`https://github.com/inria-muse/service-traceroute`

5 Planning & requirements

There are 4 weeks allocated for this project. We have created our planning in Table 1 accordingly.

Week #	**Start - End date**	**Task(s)**
Week 1	23 Sep - 28 Sep	Research related work
Week 2	29 Sep - 5 Oct	Setup lab environment Contact volunteers
Week 3	6 Oct - 12 Oct	Run and analyze experiments
Week 4	13 Oct - 19 Oct	Finalize results Write final report

Table 1: Project planning

The requirements for setting up our lab environment for this research, are up to three IPv4 and IPv6 capable switches with NAT, VLAN, routing support, and VRF. Also we will utilize the servers already provided to us to host our infrastructure.

6 Ethical implications

This research will conform to the standards set by the Netherlands Code of Conduct for Research Integrity. This means that we aim to keep our research honest, transparent, independent and scrupulous[3]. Every analysis will be performed on either servers under our control or on participants who have agreed to participate in our research project. Our service will display a disclaimer to visitors, that aims to inform them about the purpose of this research and provides the option to be excluded from participating in it. No sensitive data will be made available and we wont do any scanning of systems that don't initiate a connection with our system first. Therefore we limit the crossing of ethical and/or privacy boundaries. This still means we have to be responsible and informative since our results can have implications for networks not belonging to us.

References

[1] Ivan Morandi, Francesco Bronzino, Renata Teixeira, and Srikanth Sundaresan. *Service Traceroute: Tracing Paths of Application Flows: Methods and Protocols*, pages 116–128. 03 2019.

[2] Rob Sherwood and Neil Spring. Touring the internet in a tcp sidecar. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 339–344, New York, NY, USA, 2006. ACM.

[3] VSNU. Netherlands code of conduct for research integrity, 2018.