An Experiment with SDN-based Traffic Analysis Resistant Network (TARN) Architecture

Qing Wang, Geddings Barrineau, Junaid Julfiqar, Kuang-Ching Wang
Department of Electrical and Computer Engineering
Clemson University
Clemson, SC, USA
{qw, cbarrin, jzulfiq, kwang}@clemson.edu

Jon Oakley, Lu Yu, Richard Brooks
Department of Electrical and Computer Engineering
Clemson University
Clemson, SC, USA
{joakley, lyu, rrb}@g.clemson.edu

***Abstract*—Traffic analysis refers to methods that discover end-to-end Internet communications patterns by observing side-channels. This works even when the traffic is encrypted. Today's Internet Protocol (IP) networks are vulnerable to traffic analysis; which makes Internet censorship, man-in-the-middle (MITM) attacks and network surveillance possible. The presence of clear-text source and destination IP addresses in TCP/IP headers makes communications flows trivially easy for adversaries to detect and attack. To make traffic analysis more difficult, proxy-networks (Tor, Psiphon, Lanten, VPNs...) obscure destination IP addresses by routing traffic through one or more proxy nodes. However, proxy nodes are still being detected and blocked by nation states, among others. In addition, such solutions force users to trust intermediate proxy nodes, which sometimes execute MITM attacks. A software defined networking (SDN) based solution called TARN has been proposed to provide an end-to-end network architecture to remove the basic traffic analysis vulnerability. To resist traffic analysis, TARN binds communication sessions to randomized, short-lived, perpetually changing IPv4/IPv6 addresses. This traffic analysis resistance is achieved through the placement of a SDN at the network edge, combined with the use of BGP routing in the network's core. This paper discusses the TARN architecture in detail and its experimental evaluation on the National Science Foundation's Global Environment for Network Innovations (GENI) and the Pairing Emulated Experiments with Real Inter-domain Network Gateways (PEERING) testbeds. Results indicate end-to-end communication can be achieved over the Internet with TARN, while avoiding server IP detection and censorship.**

Keywords—Internet Security; Censorship; Traffic Analysis; OpenFlow; TARN; Software-defined Networking; SDN; Cloud Computing; Internet Architecture; GENI

I. INTRODUCTION

From its inception, the Internet was a trusted enclave - However, Internet privacy and security are not intrinsic in the IP networking architecture and protocol today, due to the reason that many untrustworthy people connected to the Internet. The well-known consequences of Internet censorship, Man-in-the-middle (MITM) attacks, and distributed denial of service (DDoS), to name a few, pose significant privacy and security problems.

One fundamental Internet security and privacy concern is unwanted traffic analysis. Traffic analysis uses packet meta data and side-channel analysis to infer end-to-end Internet communications, for both clear-text data and encrypted data. Countermeasures to traffic analysis, such as Onion Routing(Tor), Psiphon, and I2P [1,2,3], hide destination IP addresses by routing traffic through one or more proxy nodes. However, users must trust the intermediate proxy nodes, any of which could be compromised – and perform a MITM attack. In addition, all the proxy-based solutions are commonly detected and blocked by countries who are averse to the free dissemination of information. Decoy routing, a new alternative, inserts steganography information into the header of SSL packets which is recognized by "decoy routers", who re-route packets to their intended destination [4]. However, obfuscating and deploying decoy routers to avoid being systematically "bypassed" by adversaries remains an open challenge.

The aforementioned solutions mentioned are based on today's Internet Protocol(IP) architecture; however, the underlying vulnerabilities being exploited by traffic analysis are IP network sessions bound with clear-text static IP addresses in IP packet header. The proposed TARN architecture seeks an alternative way to remove these vulnerabilities. The TARN architecture propose to use a range of possible SDN solutions on the network edge and the network core to generate randomized, short-lived and perpetually changing IP prefix for a network. This makes the traffic unpredictability and evasive, in order to circumvent traffic analysis. Other SDN security solutions, such as the OpenFlow-based random host mutation (OFRHM) [5], aim to use OpenFlow to randomly mutate source host IP addresses to avoid the unwanted traffic analysis. However, OFRHM is very similar to the proxy solutions and is susceptible to being censored by blocking the destination address, while the TARN solution extends the uncertainty of randomized source and destination IP addresses across subnets and geographic borders.

To evaluate the TARN's architecture, we constructed an experiment on both the GENI[6] and PEERING testbeds[7]. GENI is a large-scale distributed testbed that gives academics and corporations access to physical and virtual network resources deployed in various geographic locations across the United States. PEERING is a research testbed that allows researchers to announce BGP prefixes to the Internet at

selected nodes in the US and EU. The experiment in GENI provides a SDN at the network edge, while PEERING implements BGP routing in the network core. The SDN implementation was done using the Floodlight OpenFlow controller, which provides server and client IPv4 address generation and end-to-end transport session maintenance. BGP routing in the network core was designed with multiple and changing IP prefixes. At this stage, the GENI testbed provides the flexibility and convenience to provision network resources at the network edge. In the future, the GENI testbed will be used to simulate the network core. The PEERING testbed connects to the Internet, where it allows us to safely exchange BGP routes and traffic with neighboring autonomous systems (AS).

II. TARN ARCHITECTURE

This section first gives an overview of the architecture of the SDN-based traffic analysis resistant network (TARN) architecture. It then introduces the main components of this architecture and the function of each.

A. Service Architecture Overview

The proposed TARN architecture, as shown in Figure 1, will allow a client to access a server in a different AS, while obfuscating traffic analysis attacker if an end-to-end communication session exits. It is worth to mentioning that the TARN architecture will potentially have multiple forms, starting with a lightweight end-to-end solution working with today's BGP Internet, and heading towards a highly programmable and morphable solution based on distributed software defined exchanges (SDX).

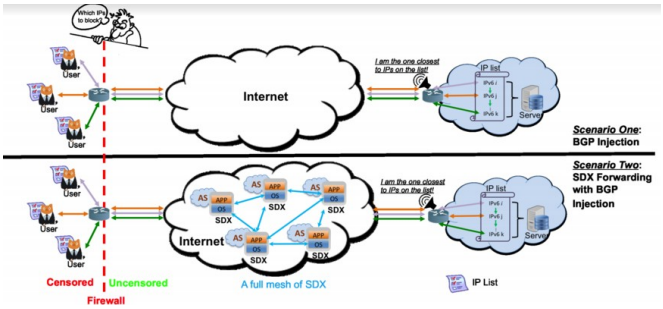

Figure. 1 TARN architecture with two potential forms

B. Functionality of Main Components

The architecture can be logically broken down to three major components: (1) the server IPv4/IPv6 addresses generation and distribution, (2) the end-to-end transport session maintenance, and (3) routing with multiple changes in IP prefixes. These three parts will be discussed below.

1) Server IP addresses generation and distribution

The objective of this part is to obfuscate traffic analysis, by constantly changing the source and destination IP addresses to multiple short-lived, randomized IP addresses. After mapping real IP addresses to random IP addresses, the real server IP address is no longer observable. This approach is particularly flexible with the vast IPv6 address space. Secure DNS techniques can be used to bootstrap this addressing scheme by securely distributing algorithms for the address "hopping" pattern.

2) End-to-End transport session mainenance

The objective of this part is to ensure the end-to-end communication is stable with server perpetually changing the IP addresses. Multiple methods here can be applied, such as Port Forwarding, or using the SDN OpenFlow Floodlight Controller (Floodlight)[8] to rewrite the headers of the packets in session. Another objective is to determine the reasonable throughput of TARN, under different network conditions.

3) Routing with multiple changes in IP prefixes

The objective of this part is to route the packet to the desired server with random, ephemeral addresses across the wide area network (WAN). This requires safely injecting BGP routes into the Internet with random and dynamic prefixes. In today's Internet, ISPs usually do not accept such practices. Therefore, we use PEERING testbed, or potentially ISPs' guidelines, to safely introduce BGP routes to the Internet, in order to determine the feasibility and usefulness of this BGP injection method. Alternatively, SDX is the mechanism we envision this fluidic IP network mapping to use in the future. Our vision is that IP network prefixes will be a dynamic network resources in a future Internet architecture, as they should be more flexible aggregated and dynamically distributed by different organizations over time.

III. TARN EXPERIMENT USING GENI TESTBED

This section first introduces the details of the experiment design using the GENI testbed. It then shows the evaluation of the experiment.

A. Experiment Design Overview

The TARN architecture will leverage a range of possible SDN solutions to achieve resistance to traffic analysis. At this stage, this resistance is achieved through the placement of a SDN at the network edge, combined with the use of BGP routing in the network core. The GENI experiment we designed to simulate this is then shown in Figure 2

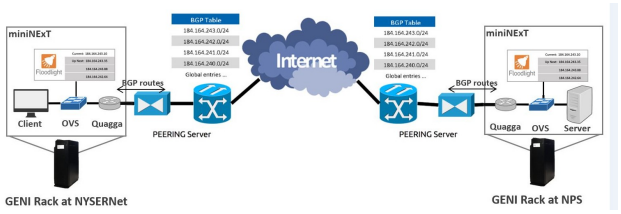

Figure. 2 GENI experiment design overview

There are two reserved GENI racks that represents the server and client side, located at NPS and NYSERNet, respectively. On each GENI rack, there is a virtual network emulator called miniNExT [9] installed. Inside each miniNExT, it includes a Floodlight manager that rewrite packet header, an OVS that interacts with Floodlight, and a Quagga router [10] running as a virtual BGP router.

Similarly to mininet, miniNExT is virtual network emulator, but it extends mininet in order to easily build complex networks. In this experiment, miniNExT uses the built-in virtual Quagga router that allows it to communicate with external BGP routers in other ASes.

To provide SDN at the network edge, we designed and deployed a Floodlight module inside each miniNExT. Based on current BGP prefix ranges, Floodlight changes the server's IP address to a random IP at a certain rate. Floodlight also leverages OpenFlow capability to perform packet header rewrite at each edge of the network, for maintaining the original communication sessions.

To provide the BGP routing in the network core, the Quagga router propagates the BGP prefixes to the Internet via PEERING. After one BGP prefix is received from the other side, it will be installed into the kernel routing table and thereby the traffic will be routable across the Internet.

B. Achieving Randomized IP addresses and Session Maintenance via SDN

In order to achieving randomized IP addresses and session maintenance, Floodlight will do two things: (1) manage a list of randomized IP addresses associated with each hosts and (2) managing flow rules for each host

1) Managing randomized IP addresses for each hosts

Each instance of Floodlight is responsible for managing a list of server hosts whose IP addresses would be randomized. Hosts are added and removed using the REST API. Each host has a real IP address as its internal IP, but there will also be random external IP address based on the current BGP prefix range. Floodlight is responsible for updating each host's external IP address at a certain rate.

It is important to mention that each Floodlight instance must be synchronized, meaning that at any given time, a single host will have the same external IP address assigned to it by every Floodlight instance. This requires all the Floodlight instance are updating at the same time and assigning the same external IP addresses to each host.

2) Managing sessions with each randomized host

After the session initiates, from each host perspective all the fields inside that communication session should stay the same. Floodlight needs to create, insert and manage flow rules that performs the packet header rewrite in order to maintain the transport layer communication sessions. Therefore, all communication sessions will remaining static with unchanging IP addresses from the host's perspective. However, from outside world perspective, the IP addresses of the hosts are always changing. There are two types of packet generated by Floodlight: IP packets and ARP packets. With each packet type, there are a pair of flows: one flow rewrites the internal IP address to external IP address, while another flow rewrites from the external IP address to internal IP address. Floodlight ensures flows are only inserted for the sessions that don't have a rewrite flow inserted yet.

C. Achieving BGP Routing in the Network Core via PEERING and miniNExT

In order to achieving BGP routing in the network core, the experiment leverages the Quagga router inside miniNExT as well as the PEERING testbed.

1) PEERING testbed

The PEERING testbed controls multiple IP prefixes, and allows clients to announce allocated BGP prefix at selected US and EU point of presences (PoPs) to the Internet. Currently, there are four IP prefixes reserved for experiment: 184.164.240.0/24, 184.164.241.0/24, 184.164.242.0/24, and 184.164.243.0/24.

2) Quagga Router in miniNExT

At each side of miniNExT, Quagga router is attached on the upstream PEERING server via PEERING's OpenVPN tunnel. PEERING runs its own Quagga software routers to establish BGP session with their BGP neighbors. Instead of running the BGP route selection process those PEERING servers provide PEERING client full control over BGP route announcements. So even our Quagga router builds a BGP peer session with PEERING Quagga routers, any BGP announcement from our Quagga router can propagate to the Internet through the PEERING server, and vice versa. The PEERING server will just relay the announcements, as long as the BGP announcement is confirmed valid by PEERING.

D. Experiment Results

The main objective for this experiment is to set up and evaluate the proposed SDN-based TARN architecture using GENI testbed. At this point, the evaluation mainly focuses on the host-to-host communication with the condition that the host external IP addresses keep changing.

More specifically, the architecture will be evaluated for the feasibility of 1) Randomized IP and Session Maintenance via SDN at network edge 2) BGP routing at network core.

To analyze 1), we run the end-to-end ping test and use tcpdump to capture a series of packets on the network edge, which is the location that randomized IP addresses mapping and session maintenance happen. To prove the experiment working, we will observe and confirm the correctness of packet header.

From each host perspective, every packet header in the session should only contain the internal IP address, since each host only knows the real source IP address and the real destination IP address. After packet go across network edge, the header will be rewritten. Therefore, from the outside world perspective, the internal host IP address will never been observed but instead of many randomized external IP addresses.

In this experiment, the internal IP address for the server is 10.0.0.1, while the internal IP address for the client is 184.164.242.100. Figure 3 and Figure 4 is the wireshark screen-shot that illustrates the internal IP distribution from host perspective, and the external IP distribution from Internet perspective. Clearly, each host only knows the internal IP address, which is 10.0.0.1 and 184.164.242.100. While from Internet perspective, there is no internal IP address but all the randomized external IP addresses are observed.

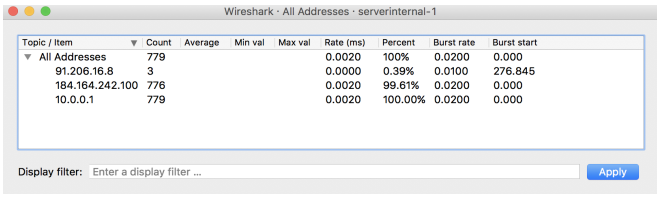

Figure 3. Internal IP distribution from host perspective

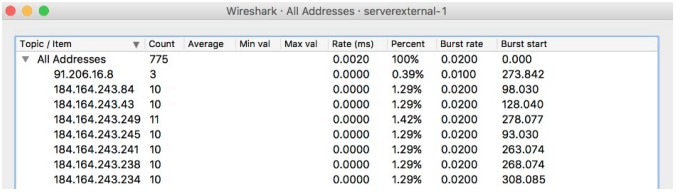

Figure 4. External IP distribution from Internet perspective

To analyze 2), it must be confirmed the BGP route is propagated to the Internet, we advertise two BGP routes and use public AT&T route servers and our miniNExT Quagga router to confirm. The advertised two BGP routes are: 184.164.243.0/24 and 184.164.242.0/24.

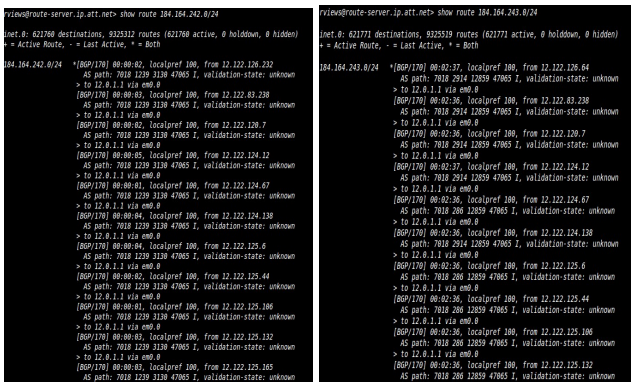

Figure 5. Advertised BGP routes 184.164.243.0/24 and 184.164.242.0/24 on AT&T route server

Above, Figure 4 shows that the two advertised routes 184.164.243.0/24 and 184.164.242.0/24 are received. The origin AS number 47065 is the master site of the PEERING testbed, which indicates the two BGP routes were advertised from our Quagga routers, as all our advertised routes will go through PEERING's master site.

Below, Figure 5 and Figure 6 show that both Quagga routers correctly receive the BGP prefixes from the other side.

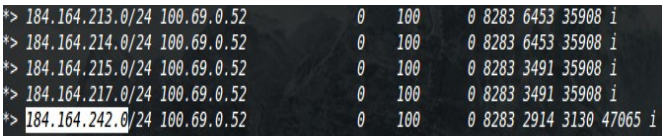

Figure 6. Server side Quagga received BGP prefix 184.164.242.0/24 announced from client side

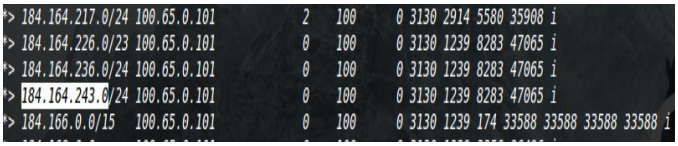

Figure 7. Client side Quagga received BGP prefix 184.164.243.0/24 announced from server side

Iç. Summary and Future Work

A. Summary

In this paper, a novel SDN-based traffic analysis resistant network architecture (TARN) is presented and evaluated as proof of concept. TARN aims to leverage a SDN solutions to obfuscate traffic patterns, preventing attackers from correlating the TARN connection with the real end-to-end connection. Particularly in this paper, an early prototype on GENI has been deployed to evaluate the core TARN architecture concepts: 1) at network edge node is randomizing IP addresses and maintaining sessions via SDN and 2) at the network core is successfully announcing new BGP routes over the wide area.

B. Future Work

TARN is sponsored by NSF award #1643020. This project was started 3 months ago, and there is still a lot work that needs to be researched and implemented. In current phase, we will continue to run this experiment and collect real network data. At the next phase we will add on IDS devices to the BGP router and play red team/blue team games to simulate different types of traffic cases and use various traffic analysis techniques to evaluate TARN's performance.

References

[1] Dingledine, R., Mathewson, N., and Syverson, P.2004. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*. 303-319

[2] Psiphon, "The Psiphon project". [online]. http://www.psiphon.ca/

[3] I2P, "The i2p network". [online]. http://www.i2p2.de/

[4] J. Karlin, D. Ellard, A. W .Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer. Decoy routing: Toward unblockable Internet communication. In *Processdings of the USENIX workshop on Free and Open Communications on the Internet*, 2011

[5] J. H. Jafarian, E. AI-Shaer, and Q. Duan. OpenFlow Random Host Mutation: Transparent Moving Target Defense using Software-Defined Networking. In *Proceedings of ACM Sigcomm HotSDN Workshop*, 2012

[6] GENI. [online]. 2016, http://www.geni.net/

[7] PEERING. [online]. 2016, https://peering.usc.edu/

[8] Floodlight, "Floodlight OpenFlow Controller" . [online]. 2016, http://www.projectfloodlight.org

[9] miniNExT. [online]. 2016, https://github.com/USC-NSL/miniNExT

[10] Quagga. [online]. 2016, http://www.nongnu.org/quagga/