Beyond the VPN: Practical Client Identity in an Internet with Widespread IP Address Sharing

Yu Liu and Craig A. Shue
Worcester Polytechnic Institute
{yliu25, cshue}@wpi.edu

***Abstract*—To support remote employees, organizations often use virtual private networks (VPNs) to provide confidential and authenticated tunnels between the organization's networks and the employees' systems. With widespread end-to-end application-layer encryption and authentication, the cryptographic features of VPNs are often redundant. However, many organizations still rely upon VPNs. We examine the motivations and limitations associated with VPNs and find that VPNs are often used to simplify access control and filtering for enterprise services.**

To avoid limitations associated with VPNs, we propose an approach that allows straightforward filtering. Our approach provides evidence a remote user belongs in a network, despite the address sharing present in tools like Carrier-Grade Network Address Translation. We preserve simple access control and eliminate the need for VPN servers, redundant cryptography, and VPN packet headers overheads. The approach is incrementally deployable and provides a second factor for authenticating users and systems while minimizing performance overheads.

***Index Terms*—Virtual private networks, access control, software-defined networking, residential networks, NAT**

I. Introduction

Virtual private network (VPN) tools allow organizations to manage remote users' access as if they are local. VPN was designed to ensure confidentiality, integrity, and authenticity of remote user communication. However, in recent years, the deployment of end-to-end cryptography has grown substantially, with over 90% of web servers supporting TLS [8]. Since confidentiality, integrity, and authenticity can be reasonably assured at the application layer, the VPN's cryptography may often be redundant. Further, it comes at a cost: 1) VPN servers act as an aggregation point that can become a bottleneck [4], 2) a greater portion of each network packet is used for headers for the cryptographic protocols, and 3) VPN licenses for remote users can be expensive for organizations [3].

Enterprises are often motivated to deploy VPNs in order to: 1) protect data confidentiality and authenticity, 2) manage communication at the remote endpoint, and 3) simplify access control. With redundant cryptography, the first goal is decreasingly important. The second goal, as we discuss in Section III, can be better achieved with endpoint filtering. We explore the remaining goal, access control, in detail in this work.

Within the network, an organization may use network address translation (NAT) to assign hosts and devices to private, unroutable address space so that the infrastructure cannot be reached by outsiders without going through NAT devices [22]. Such network addresses can be used as unique identifier in allow-lists for intranet service management, like web, email and firewalls [2], [18]. However, such address filtering is impractical with remote users, since Internet Service Providers (ISPs) often use dynamic addresses for their customers. Some ISPs have adopted carrier-grade network address translation ("carrier-grade NAT" or CGN) to dynamically share limited IPv4 address space. CGN was used in 92% of cellular networks [23] by 2016. Some providers estimate that 30 million home networks will be affected by CGN with the deployment of 5G networks.

While organizations may wish to use NAT with VPN servers to enable address-based filter, the VPN overheads may be a challenge. Ideally, organizations would have a network-level identifier that would allow quick and easy validation of a remotely-connecting end user. Such identifiers could serve as a "first-factor" authenticator that provides evidence of the connecting machine's or network's likely legitimacy. This factor can then be combined with other robust authentication factors, such as application-layer credentials, on the server endpoint. Importantly, the end-user and the organization must both assent to use of the identifier to avoid misuse.

In this work, we explore a practical and deployable approach to allow end-users to create dynamic network-level factors for access control. Our contributions include:

- **Dynamic Identifier Insertion:** Using software-defined networking (SDN) techniques, we allow endpoints and residential routers to insert authentication identifiers into traffic in an application-agnostic manner. Organizations may use these identifiers to for lightweight access control.
- **Gateway and Endpoint Validation:** Using `iptables`, we allow organizations to deploy validater conveniently, which provides functionality similar to a VPN without creating unnecessary bottlenecks.

II. Background and Related Work

We provide background on CGN, software-defined networking, user identity, and identifier encoding.

A. Carrier-Grade NAT (CGN) and Address Sharing

To help with the scarcity of IPv4 addresses, providers use CGN to share addresses across users [23]. CGN typically combines network addresses and transport layer ports to map traffic to the appropriate end-user. Some guides recommend 30,517 public IP address for every 1,000,000 subscribers [20]. Richter et al. [23] found that while only 13.3% of non-cellular ASes use CGNs, 92% of cellular networks use them [23]. In

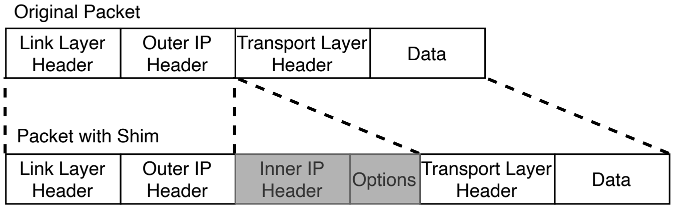

Fig. 1. IP-in-IP packet format.

their 5G cellular deployment plans, some providers estimate they will serve 30 million home networks [24]. Address sharing is widespread on the Internet and is expected to grow.

B. Software-Defined Networking (SDN)

The SDN paradigm uses a centralized controller to separate the data plane and control plane of network traffic. The OpenFlow protocol [17] allows the controller to alter data structures in switches and routers to enable inspection and arbitrary forwarding of packets. OpenFlow can be enabled in software bridges on endpoints via Open vSwitch [15]. With OpenFlow, a controller can typically inspect or alter the first few packets in a network flow and then order switches to cache appropriate rules through a `FLOW_MOD` order for handling subsequent packets in a flow. This can minimize the performance overheads.

C. IP Addresses and Host Identity

Address sharing affects a wide range of public IP address-based applications, such as firewall policies. IP reputation is used by major email providers, in tools such as Microsoft's SmartScreen [18], to assess risk associated with incoming email messages [1]. Such tools may mistakenly assume that IP addresses change infrequently and are unlikely to be shared. This leads to false negatives when attackers move across IP addresses and false positives when innocent people happen to use an IP address previously involved in an attack [9].

Komu et al. [14] surveyed Internet architecture efforts that aim to split the "locator" functionality from network addresses from the "identifier" associated with the system. Methods like HIP [19] and MILSA [21] propose to assign persistent identities to each host and use network address solely for routing and forwarding. The required mapping between identifiers and addresses requires infrastructure or OS changes, which complicate deployment. Further, ISPs have used "super cookies" for persistent identity [6] at the cost of user privacy [12].

D. Mechanisms to Encode Application-Agnostic Identifiers

We use an IP-in-IP shim to add options in a backwards-compatible, standardized manner, as shown in Figure 1. We insert a second IP header in front of the original transport layer header. The outer IP address is used as the locator and the inner IP header conveys identifier data. While we could re-purpose header fields inside the inner IP header to encode identifiers, this may cause intermediate routers that inspect the inner header to drop the packets if they are considered malformed. We thus leave these fields unchanged.

III. Motivations for VPN Deployments

Organization commonly use VPNs with their employees to: 1) protect packet confidentiality, integrity, and authenticity via cryptography, 2) control communication to remote systems, and 3) simplify access control by signalling a host's "insider" status [11], [16].

Web protocols make up the majority of Internet traffic, and recent surveys show that over 90% of web traffic is now protected by TLS [5]. TLS is common in business application communication, including web and email traffic. Other protocols, such as remote desktop tools, file transfer, secure shell and network printing, commonly use application-layer cryptography. A significant portion of communication between remote users and an organization's network already have application-layer security assurances.

Some legacy protocols or devices may not support cryptography. However, organizations may use the reverse proxy model [7] to protect such devices without requiring VPNs.

IV. Approach: Indicating Authenticity Validation

Inspired by Kerberos [13] and HTTP cookies, we explore a token-based design to provide evidence showing a client is likely an authenticated user in the system.

A. Design Goal: Evidence Supporting Legitimacy

Our goal is to re-create the weak connection evidence for an organization's servers in a lighter weight manner. We want to present the server with evidence that an end-user interacted with an authentication server at the organization and is likely to be legitimate. We aim to simplify first-pass network connection filtering through a simple authentication token, while leaving robust user identification to the application layer.

Our approach is designed to be effective against "off-path" adversaries, such as other clients that happen to share a CGN gateway and are multiplexed onto the same IP address. The authenticator value is constructed to make it difficult for an adversary to guess the value and impersonate the client. However, the approach is not designed to be robust against an "on-path" adversary, such as a network provider. Such an adversary would be able to inspect or alter the IP-in-IP shim layer and discover the identifiers. Further, the shim is only inserted in the first packet in the flow, so an on-path adversary could simply hijack the flow for malicious purposes after the initial verification. However, this threat model is consistent with our design goals of keeping robust authentication at the application layer.

B. Leveraging Authentication Servers

As shown in Figure 2, when the client initiates an authentication request, the first packet of the flow is elevated to the SDN controller. The controller learns about the authentication server and its support of our protocol through DNS records, which are configured by the organization. The controller

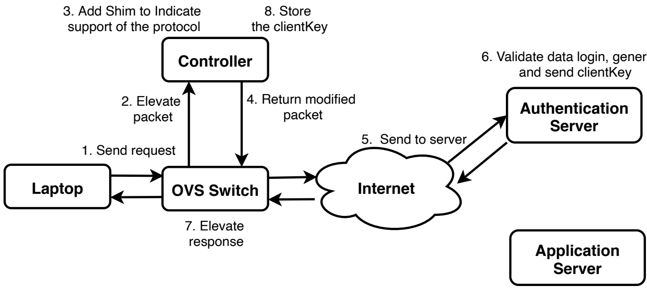

Fig. 2. The key transfer from the authentication server to the OpenFlow controller.

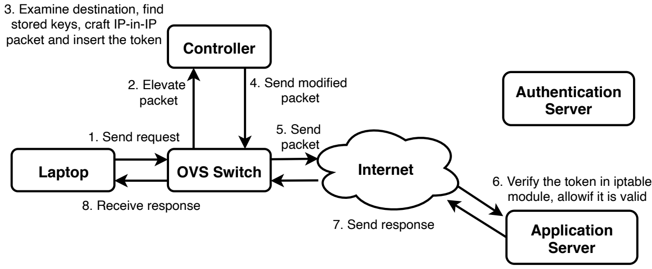

Fig. 3. The process for the client to authenticate to the application server.

signals the client's support via an IP-in-IP shim. When the authentication server replies, it includes a client key that the client can use to authenticate itself using an HMAC derived using a pre-shared secret between the authentication server and application server.

The authentication server communicates the key and identifier by crafting an IP-in-IP packet. The inner IP header includes an option field in which both the client key and a unique identifier are encoded. The OpenFlow switch simply elevates any packet with IP-in-IP shims to the controller for review, allowing the controller to obtain the key and identifier.

C. Using OpenFlow to Manage Tokens

As shown in Figure 3, when a client initiates a query to the application server, the OpenFlow switch elevates the request to the OpenFlow controller. The controller consults its database and user preferences, determines that a token is needed. Then the controller creates an IP-in-IP shim. In the inner header, it creates an IP option that contains a token that is an HMAC derived from the unique identifier, client key, and a nonce. The controller sends this modified packet back to the switch for transmission using an OpenFlow PACKET_OUT message.

The application server parses the IP-in-IP message, validates the token, and removes it before delivering it to the actual destination application. To do this, we develop open source iptables modules using the Xtables-Addons framework [10]. We develop a match module to intercept the packets of our protocol and verify the token. If the result is positive, the second target module removes the inner IP header and decapsulates the IP-in-IP packet so the application can process it. With these modules, any network flows with unverified tokens will be denied access.

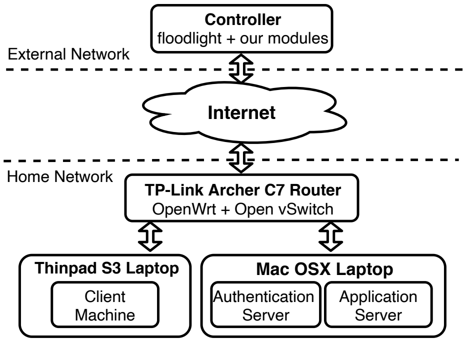

Fig. 4. Our experiment architecture.

V. Implementation

We implement our system in a home network, shown in Figure 4, to evaluate its security and performance. We examine the required modifications for a regular client, an authentication server (e.g., an identity provider), an application server (e.g., a relying party), and an SDN controller.

We create our client host in a virtual machine (VM) on a physical machine. On a different physical machine, we run the application server and the authentication server as VMs. For our SDN gateway, we use a TP-Link Archer C7 router running OpenWrt and an Open vSwitch module. The two physical machines are connected via two LAN ports of the router. The VMs bridge their network interfaces and receive their DHCP leases from the router. The router and two physical machines are located in a residential LAN. The controller runs on a remote university campus server, emulating the use of a remote third-party SDN service provider. We implement our SDN controller and module using the Floodlight SDN framework.

VI. Evaluation: Security and Performance

To evaluate the security of the approach, we create new connections from the client machine to the application server. The application server runs our iptables modules locally to allow validated packets with a default deny rule to drop any new flows that are not approved by our match module.

In our first scenario, we disable the OpenFlow controller module, causing the client to send packets without a token. All such attempts were blocked by the default deny rule. We further enable our OpenFlow module that intercepted only the first packet in each flow to insert the properly constructed header but with an invalid token. Likewise, all such flows were properly blocked. Finally, when we use the OpenFlow module to construct a proper header and a valid token value all such flows were properly matched and authorized. In each scenario, we ran 20 trials.

To evaluate the performance of our key transfer module, we transmit a TCP SYN packet to a port without an associated application server, resulting in a TCP RST packet that refuses the connection. This simple exchange allows us to monitor any overheads at the OpenFlow controller and iptables modules to signal support for the protocol, encode keys into packets, and extract those keys.

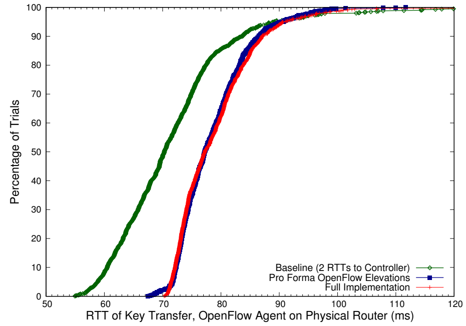

Fig. 5. Key transfer between client and authentication server across 1,000 trials, with the TP-Link router executing the OpenFlow agent.

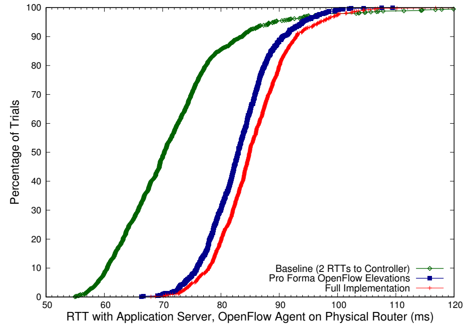

Fig. 6. In key validation, the client interact with application server in 1,000 trials. The OpenFlow agent runs on the physical TP-Link router.

During the key transfer at the authentication server, the client sends 1,000 TCP requests and measures the round-trip time (RTT) that includes all the overheads. Two elevations to the OpenFlow controller are required. In Figure 5, the leftmost (green) line indicates baseline of two RTTs with the controller. The 90th percentile is 83ms. The middle (blue) line shows the measured end-to-end RTT for the client to the server using *pro forma* OpenFlow elevations that omit alterations. The 90th percentile of the RTT is 86ms. The 90th percentile of our full implementation (the rightmost, red line) rises to 87ms.

To measure the performance evaluation of the key validation associated with the application server, the client sends a UDP packet and measures the RTT in 1000 trials. In each trial, the packet is elevated to the controller twice: once to perform the IP-in-IP encapsulation and once to handle the UDP server's response. In Figure 6, the baseline of two RTTs with the controller remains the same (green, leftmost line) with a 90th percentile RTT of 83ms. In the pro forma scenario (blue, middle line), the 90th percentile RTT is 90ms. In the full implementation scenario (rightmost, red line), the 90th percentile RTT is 93ms.

VII. Conclusion

We explore an alternative to VPNs for simplified access control filtering using a shared identifier. This allows endpoint servers to pre-filter traffic, achieving the same goals while reducing bottlenecks, extra server infrastructure, redundant cryptography, packet header overheads, and complexity. We proposed an SDN-based architecture to facilitate user-controlled persistent identity. We used `iptables` modules to implement the approach and the results show that our method is effective, lightweight, and incrementally deployable.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1651540.

References

[1] Amazon, Inc., "Amazon SES IP blocklist FAQs," https://docs.aws.amazon.com/ses/latest/DeveloperGuide/blocklists.html, 2019.

[2] Apache, "Access control," https://httpd.apache.org/docs/2.4/howto/access.html, 2020.

[3] D. Athow, "7 best business VPN solutions 2020," https://www.techradar.com/news/best-vpn-for-business-our-5-top-choices, 2020.

[4] M. Cooney, "Coronavirus challenges remote networking," https://www.networkworld.com/article/3532440/coronavirus-challenges-remote-networking.html, 2020.

[5] C. Cullen, "Sandvine releases 2019 global internet phenomena report," https://www.sandvine.com/press-releases/sandvine-releases-2019-global-internet-phenomena-report, 2019.

[6] A. C. Estes, "The dangers of supercookies," https://www.theatlantic.com/technology/archive/2011/08/dangers-supercookies/354297/, 2011.

[7] F5, Inc., "What is a reverse proxy server?" https://www.nginx.com/resources/glossary/reverse-proxy-server/.

[8] Google, "Https usage in chrome worldwide," https://transparencyreport.google.com/https/overview?hl=en&time_os_region=chrome-usage:1;series:time;groupby:os&lu=load_os_region&load_os_region=chrome-usage:1;series:page-load;groupby:os, 2020.

[9] K. Hill, "Cloudflare blocking my IP?" https://community.cloudflare.com/t/cloudflare-blocking-my-ip/65453, 2019.

[10] N. Jamili, "xtables-addons," https://github.com/nawawi/xtables-addons, 2020.

[11] A. G. Johansen, "10 benefits of VPN you might not know about," https://us.norton.com/internetsecurity-privacy-benefits-of-vpn.html, 2020.

[12] J. Kastrenakes, "FCC fines verizon 1.35 million over 'supercookie' tracking," https://www.theverge.com/2016/3/7/11173010/verizon-supercookie-fine-1-3-million-fcc, 2016.

[13] J. Kohl and C. Neuman, "The Kerberos network authentication service (V5)," IETF RFC 1510, 1993.

[14] M. Komu, M. Sethi, and N. Beijar, "A survey of identifier–locator split addressing architectures," *Computer Science Review*, vol. 17, pp. 25–42, 2015.

[15] Linux Foundation, "Open vSwitch," https://www.openvswitch.org, 2016.

[16] T. McCue, "Benefits of a VPN," https://www.forbes.com/sites/tjmccue/2019/06/20/benefits-of-a-vpn/#149611632466, 2019.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[18] Microsoft Corporation, "Troubleshooting," https://sendersupport.olc.protection.outlook.com/pm/troubleshooting.aspx, 2018.

[19] R. Moskowitz, P. Nikander, P. Jokela *et al.*, "Host identity protocol (HIP) architecture," IETF RFC 4423, 2006.

[20] K. Nishizuka, "Carrier-grade-NAT (CGN) deployment considerations," IETF Draft, https://tools.ietf.org/id/draft-nishizuka-cgn-deployment-considerations-00.html, 2013.

[21] J. Pan, S. Paul, R. Jain, and M. Bowman, "MILSA: a mobility and multihoming supporting identifier locator split architecture for naming in the next generation internet," in *IEEE GLOBECOM*, 2008.

[22] Y. Rekhter, B. Moskowitz, D. Karrenberg, and G. de Groot, "Address allocation for private internets," IETF RFC 1597 https://tools.ietf.org/html/rfc1597, 1995.

[23] P. Richter, F. Wohlfart, N. Vallina-Rodriguez, M. Allman, R. Bush, A. Feldmann, C. Kreibich, N. Weaver, and V. Paxson, "A multi-perspective analysis of carrier-grade NAT deployment," in *Internet Measurement Conference*. ACM, 2016, pp. 215–229.

[24] B. Varettoni, "Verizon to launch 5G residential broadband services in up to 5 markets in 2018," https://www.verizon.com/about/news/verizon-launch-5g-residential-broadband-services-5-markets-2018, 2017.