PROVIDE: Hiding from Automated Network Scans with Proofs of Identity

William Koch
Department of Computer Science
Boston University
Boston, MA 02215
wfkoch@bu.edu

Azer Bestavros
Department of Computer Science
Boston University
Boston, MA 02215
best@bu.edu

***Abstract*—Network scanners are a valuable tool for researchers and administrators, however they are also used by malicious actors to identify vulnerable hosts on a network. Upon the disclosure of a security vulnerability, scans are launched within hours. These opportunistic attackers enumerate blocks of IP addresses in hope of discovering an exploitable host. Fortunately, defensive measures such as port knocking protocols (PKPs) allow a service to remain stealth to unauthorized IP addresses. The service is revealed only when a client includes a special authentication token (AT) in the IP/TCP header. However this AT is generated from a secret shared between the clients/servers and distributed manually to each endpoint. As a result, these defense measures have failed to be widely adopted by other protocols such as HTTP/S due to challenges in distributing the shared secrets.**

In this paper we propose a scalable solution to this problem for services accessed by domain name. We make the following observation: automated network scanners access servers by IP address, while legitimate clients access the server by name. Therefore a service should only reveal itself to clients who know its name. Based on this principal, we have created a proof of the verifier's identity (a.k.a. PROVIDE) protocol that allows a prover (legitimate user) to convince a verifier (service) that it is knowledgeable of the verifier's identity. We present a PROVIDE implementation using a PKP and DNS (PKP+DNS) that uses DNS TXT records to distribute identification tokens (IDT) while DNS PTR records for the service's domain name are prohibited to prevent reverse DNS lookups. Clients are modified to make an additional DNS TXT query to obtain the IDT which is used by the PKP to generate an AT. The inclusion of an AT in the packet header, generated from the DNS TXT query, is proof the client knows the service's identity. We analyze the effectiveness of this mechanism with respect to brute force attempts for various strength ATs and discuss practical considerations.

I. Introduction

The Internet is constantly being scanned for vulnerable hosts [1]. These scans originate from automated network scanners that send the same probe for a given range of IP addresses, also commonly referred to as "horizontal scanning". A probe can be used to identify open ports, as well as the host's operating system and names of network services. Horizontal scanning is typically conducted by opportunistic attackers, hoping that if they scan enough IP addresses their odds of finding a vulnerable host will increase. For example, a new ransomware family known as SamSam, targets and infects unpatched JBoss web servers allowing the attacker to pivot and move laterally in a corporate network [2]. Additionally, the infamous Sality botnet uses strategic horizontal scanning of the entire IPv4 address space to recruit new bots [3].

The severity of this threat is further compounded by the speeds and convenience at which the entire IPv4 address space can be scanned. Off-the-shelf high-speed scanners such as ZMap [4] and Masscan [5], combined with virtual machines with 10Gbps links available for rent, e.g., Amazon EC2 instances with enhanced networking,[1] would allow anyone with a computer and a few dollars to scan the Internet in minutes. Furthermore, it has been observed that scans are launched within hours once new security vulnerabilities have been disclosed [6]. Consequently, the attacker has a drastic advantage over the server administrator.

Although it could be argued that there are some legitimate and useful applications of Internet scanners (e.g., by researchers or network security administrators), given the current landscape, we are in dire need of a method to hide from scanners. One such defense to prevent unauthorized connections to a server is known as port knocking. Generally speaking, a port knocking protocol (PKP) is a method for communicating information across closed ports [7] [8]. It is typically used to dynamically alter a firewall to allow authorized packets access to a particular service. Initially the server's firewall is configured to reject all incoming connections. Upon being scanned, all ports will appear closed. The port becomes visible only when a client sends a packet containing an authentication token (AT) known also to the server. The server continuously monitors incoming traffic and alters the firewall to accept a connection when the correct AT is observed.

Currently, there are dozens of port knocking implementations [9], used largely to hide private services such as SSH in which the server is knowledgeable of the clients. As a result, they have yet to be widely adopted by other protocols. One explanation is the non-standardized way for distributing shared keys to derive the ATs. Unfortunately this prohibits the use of PKPs with open access protocols such as HTTP/S, DNS, and anonymous FTP which are in the top ten most popular services on the Internet [4].

We observe that horizontal scanning is performed by ac-

[1] EC2 Instance Types, https://aws.amazon.com/ec2/instance-types/

cessing a server by its IP address, whereas legitimate users typically access a server by its domain name. Therefore, we postulate that a server should only be visible to clients who know the server's name.

Based on this concept, we have created a proof of the verifier's identity (a.k.a. PROVIDE) protocol that allows a prover (legitimate user) to convince a verifier (service) that it is knowledgeable of the verifier's identity. Periodically, the verifier publishes an identification token (IDT) to a trusted shared directory, indexed by its identity. If and only if the prover is knowledgeable of the verifier's identity can it retrieve the IDT. Thus when the IDT is presented to the verifier, it provides proof they must know the verifier's identity.

We present a PROVIDE implementation using a PKP and DNS(SEC) (PKP+DNS). The implementation prohibits reverse DNS (rDNS) lookups, thereby forcing the client to know the domain to obtain the IDT. The PKP+DNS implementation allows any service accessed by a domain name, and not reliant on rDNS lookups, to be converted to a stealth service and hide from IP-based horizontal scans. Although PKPs require modifications to the client, consequently making global adoption challenging, PKP+DNS can be deployed in enterprise networks immediately to protect from internal malicious scans. We analyze the likelihood of a successful brute force attack on the PKP and show that a mere a 16-bit AT would make horizontal scanning impractical. Finally we discuss practical considerations and the impact it will have on next generation scanners.

II. Design Space

In this section we provide details of the attacker we wish to defend against, and the defenders goals.

A. Threat Model

We consider an opportunistic adversary, performing malicious horizontal scans by IP address to discover vulnerable servers. The adversary does not target any one specific server, instead they scan a large number of IP addresses to increase their odds of finding vulnerabilities. Adversaries are an endpoints and traffic does not flow through them. Additionally they are not passively monitoring network traffic. We also assume that such adversaries are reactive, and can promptly initiate a scan when a new security vulnerabilities is disclosed.

The network scanner used by an adversary is a software program, either off-the-self (e.g., ZMap [4], Masscan [5], Nmap [10]) or a custom home-brewed solution. A scan targets a set of ports for a given range of IP addresses. The scanners are capable of high-speed transmission rates, equivalent to the fastest available Internet port scanners, capable of reaching speeds up to 14.23 million packets per second (Mpps) [11]. Additionally, the network from which the scan originates has the bandwidth capable to reach such speeds.

B. Goals

Our proposed PROVIDE protocol and implementation have the following goals for the defined threat: 1) Eliminate a server's visible footprint from IP-based network scanners. 2) Provide a key distribution solution for existing PKPs. 3) Allow all services accessed by domain name to be converted to a stealth service. 4) Establish a binding between a domain name and IP address in a packet header.

III. Design

Often a party must prove its identity to a verifier to establish trust. However it is also desired to have a method to prove the *inverse*, in which the party must prove to the verifier they know the *verifier's* identity. In the following section we propose a proof of the verifier's identity (a.k.a. PROVIDE) protocol that allows the party to prove to a verifier, in a single message that they know the verifier's identity, without revealing that identity in the message. We use the protocol to hide network services from IP-based network scanners, by forcing the client (prover) to include a proof of identity in each request. If the proof does not verify, the packet is rejected and the service appears to not exist. We first introduce a general definition of the generic PROVIDE protocol specification, and then present an implementation using a PKP and DNS.

A. Proof of the Verifier's Identity

Definition 1. A proof of the verifier's identity (PROVIDE) is a two-party non-interactive protocol where a prover (P), succeeds in convincing a verifier (V) it knows the verifier's identity (ID), without actually revealing it. Both parties have read access to a trusted shared directory (D) that provides identification token (IDT) lookups for a given identity $IDT \leftarrow D[ID]$. Furthermore, the verifier has write access for its own identity, allowing it to generate and publish IDTs, $D[ID] \leftarrow IDT$. Each new IDT generated is unpredictable and indistinguishable from the previous. If the prover is knowledgeable of the verifier's identity, it queries the directory for the IDT and sends a representation of this IDT to the verifier as proof of identity. PROVIDE has the following properties:

1) **Deterministic completeness**: The verifier will accept all correct proofs.
2) **Probabilistic soundness**: The verifier will reject all incorrect proofs. However there is configurable, non-zero probability the verifier will accept a proof without the prover having prior knowledge of their identity. For example, the correct proof is guessed by brute force.
3) **Identity remains confidential**: When the prover queries the directory, and submits proof to the verifier, the verifier's identity is not revealed.
4) **Non-interactive**: The verifier cannot interact with the prover until the proof has been verified.

The prover is defined by two parameters (ID, f) representing a *priori* knowledge of the verifier's identity and a shared transform function. The transform function $f(IDT)$ allows the prover to transform the IDT into a different encoding before being sent to the verifier. The output of the transform function is also referred to as the authentication token (AT) and may be as simple as the identity function. The verifier is defined

by three parameters (ID, f, λ). The IDTs have a limited life and are replaced at a rate λ. The PROVIDE protocol for an instance in time is defined as follows:

1. At time t, $V(ID, f, \lambda)$ generates an IDT and updates D, $D[ID] \leftarrow IDT$.
2. At time $t + i \leq t + \frac{1}{\lambda}$
   2.a. $P(ID, f)$ makes the query $IDT' \leftarrow D[ID]$.
   2.b. P sends $AT = f(IDT')$ to V.
   2.c. V accepts if $f(IDT) = AT$

B. A PROVIDE Protocol Implementation

In this section we present a PROVIDE protocol implementation built using a PKP, and DNS (PKP+DNS) to create a stealth service. The implementation consists of a server (the verifier), a client (the prover) and a name server (the shared directory). This design is intended to be used with an existing PKP implementation which uses an IDT as the shared key distributed with DNS. We first make the following assumptions:

- **Clients access the service by domain name.** This implementation uses DNS to distribute the IDTs. If the service is accessed by IP address, this defense will not be applicable.
- **Reverse DNS lookups for the server's domain do not exist.** Reverse DNS lookups (rDNS) allow a domain name to be resolved from an IP address through DNS PTR records. The soundness property cannot be fulfilled for a domain name containing PTR records, as the identity can be trivially learned from the IP address without the client having prior knowledge of the identity. Services dependent on rDNS would not be suitable for this implementation. For example, rDNS is commonly used by mail servers to prevent spam.

An overview is illustrated in Figure 1. The details of the client, server, and name server are described below.

1) Name server: DNS TXT records associate arbitrary ASCII text with a domain name and are used to store IDTs without any modifications to the DNS specification. According to RFC 1464 [12], the general syntax for a DNS TXT record is,

```
<owner> <class> <ttl> TXT "<name>=<value>"
```

The owner specifies who owns the record. The class adds another dimension to a DNS record, however it is typically set to `IN` for Internet. The TTL (time-to-live) specifies the number of seconds until the record expires. Lastly we have the data string associated with the TXT record. Ultimately it is up to the application utilizing the TXT record to define how to interpret and decode the data string, however it is recommended to represent the data in (name, value) pairs. The max length of the string is 255 characters [13], however multiple strings can be associated with a single record and will be concatenate by the end host.

The IDT is stored in the TXT record data string, accessed by the name `idt`. Furthermore, the TTL is set to $\frac{1}{\lambda}$ which indicates the number of seconds the IDT will be valid for. The λ parameter provides a trade-off between effectiveness and performance overhead. As λ increases, it will reduce time IDTs can be shared and time to brute force ATs. However, this will result in an increase in overhead for the entire system as IDTs will be generated faster and the name server will be queried more frequently. Additional configuration data specific to the PKP may also be included in the TXT record to inform the client how to authenticate, e.g., providing parameters to the transform function.

In the domains zone file, the A and AAAA records associates a domain name to a IPv4 and IPv6 address, respectively. It is common for a domain name to be assigned multiple IP addresses as a method of load balancing or redundancy. Each server, at a separate IP address, is responsible for publishing their own IDT. Therefore, there must be a TXT record for each address record. A domain label is created from the human-readable IP address notation, by replacing the periods with dashes. This label is appended to the domain and set as the owner of the TXT record. Below is an example TXT record for a server with IP address `1.2.3.4` and domain `example.com`. The IDT is set to expire after a TTL of $\frac{1}{\lambda} = 120$ seconds.

```
1-2-3-4.example.com. IN 120 TXT idt=48F10
```

The name server is configured to support dynamic DNS (DDNS) [14] in order for the TXT records to be automatically updated.

2) Server: The server has three components to support the stealth service, namely: a firewall provides dynamic access control, a PKP server authorizes clients with verified packets, and an IDT publisher. At a rate λ, the IDT publisher generates a new IDT, and updates the TXT record in zone file via DDNS. One method of generating IDTs is to use time based one-time passwords (TOTP) [15]. The PKP server is notified of the new IDT, so authentication reflects the update. The PKP server continuously monitors incoming traffic, if a packet contains a valid AT, the firewall is modified to accept the packet, otherwise the packet is dropped and the server appears to be hidden (i.e., the scanner in Figure 1). In the case of virtual servers, where multiple domains can be associated with a single IP address, IDTs are generated and published for each domain, and the PKP server is configured to verify all IDTs.

3) Client: The client requires two components to access a stealth service, an IDT subscriber, and a PKP client. When a network request with a domain name is made, its IP address is first resolved. Next, the IDT subscriber sends a TXT query to the subdomain equal to the formatted IP address to obtain the IDT. In order to fulfill the confidential identity property, DNS traffic can be encrypted. The PKP client takes as input the IDT, performs a transformation to create the AT. The AT is inserted into the packet header and sent to the server.

IV. Analysis

In this section we analyze a PKPs resistance to brute force attacks. The brute force attack tries all possible AT

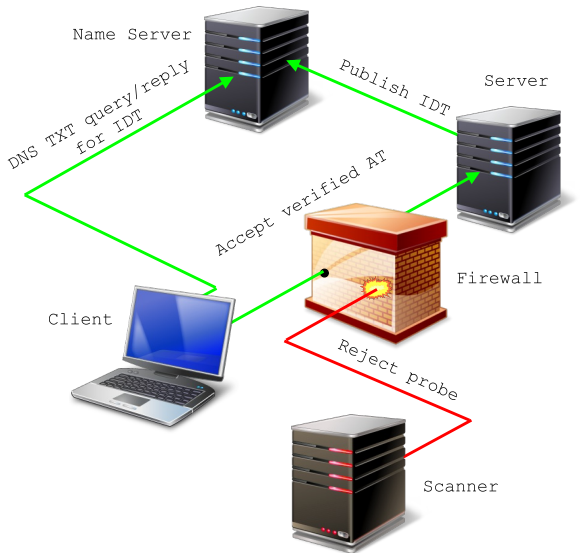

Fig. 1: Overview of PKP+DNS implementation of PROVIDE.

combinations until the correct one is guessed. We investigate the expected time for the attack to succeed for various 2^k size search spaces and different attack rates μ. Based on the findings we are able to reason about the AT length and the rate λ the IDT is updated in PKP+DNS.

We assume ATs are k-bit strings, randomly sampled from a uniform distribution. To model a brute force attack, we first select an AT from the $N = 2^k$ possibilities, and then the attacker randomly samples k-bit strings, without replacement, until the AT is selected. We define the random variable X as the number of attempts it takes to select the AT. Deriving the expected number of attempts until success is straightforward,

$$E[X] = \frac{1}{N}\sum_{i=1}^{N} i = \frac{N+1}{2} \tag{1}$$

while the expected time to brute force is $E[X]/\mu$. We calculate the expected time for various packet transmission rates and AT lengths. Our analysis results are shown in Figure 2. To date, the fastest reported network scanner is an enhanced version of ZMap, scanning at 14.23 million packets per second (Mpps) and scanning the IPv4 address space in 4m 29s [11]. For $k = 16$ (the number of bits to encode an AT in a destination port number) it would take an expected $2.18ms$ to succeed in a brute force attack at these speeds. Although this slowdown would not deter a targeted attack, it would increase the time of a single IPv4 horizontal scan to 97 days, if all end hosts deployed this defense. Furthermore, in practice, this would be a lower bound as the end host can rate limit requests. At $k = 64$, targeted attacks are no longer feasible.

An increase in scanning time of this magnitude would drastically reduce the effectiveness of horizontal scanning as a means to identity vulnerable machines. Additionally, the surge in traffic would be evidence of malicious activities. In reaction, there may be an attempt to create an alternative rDNS to lookup IDTs by IP address. As a moving target defense, the update rate, λ, limits the time the IDTs could be shared.

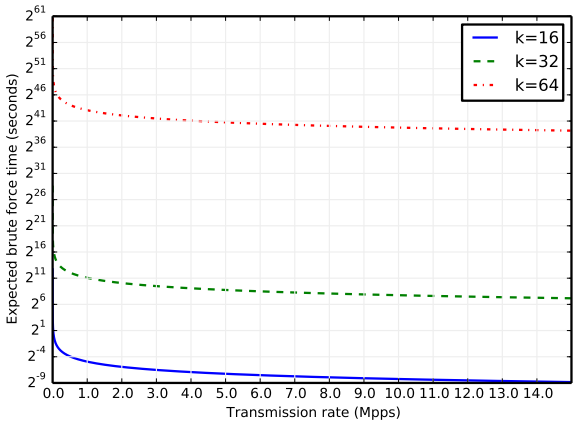

Fig. 2: Expected seconds to brute force AT for a given transmission rate in million packets per second (Mpps).

Ideally $\frac{1}{\lambda} << t$, for a scan time t, such that the IDT will be stale before scanned. However if λ is too high it will put strain on the name server.

V. Discussion

A. Deployment

Many challenges arise for global adoption of stealth services and the development of a standard. This work has addressed the key distribution problem, providing a solution for all services accessed by domain name, regardless of protocol. However, perhaps a more significant obstacle, is a standardized method for encoding ATs in packet headers and client side support. Previous literature [16] [17] has proposed several methods for encoding ATs in a packet headers. The closest we've come to a standard is a recent RFC information draft for TCP Stealth, documenting a method to send an AT as the initial sequence number (ISN) [8]. We believe an ideal encoding scheme should have the following properties,

- **As soon as possible (ASAP) verification**: Unauthorized packets should be rejected as early as possible to reduce attack surface and decrease server side processing.
- **Protocol agnostic**: The encoding scheme should not be limited to a specific protocol whether UDP or TCP.
- **Authentication for each packet**: Each packet received from a client should be authenticated.
- **Scalable**: As network speeds and transmission rates increase, so should the strength of the AT.
- **Low overhead**: The processing required to verify the AT should be minimal and not introduce a denial of service (DoS) vector.
- **Preserved during transit**: Routers, and middleboxes shall not modify the AT.

However TCP Stealth [8] fails to fulfill the first four properties. It is exclusive to TCP, and the ISN encoding prohibits each packet from being authenticated, as subsequent sequence numbers for the connection are derived from the ISN and the bytes transferred [18]. IP options is the only encoding that satisfies these properties. IPv4 options provides up to 40 bytes for the AT [19] and an arbitrary size with the introduction of

extension headers in IPv6 [20]. However as past literature has discovered [21], approximately half of the Internet routes drop packets with options. One explanation for this behavior is that processing IP options is an expensive task and is susceptible to a DoS attack [22].

Although the aforementioned challenges may delay adoption of stealth service in the open Internet, enterprise networks do not suffer from these limitations. Enterprise networks have full control over their internal network infrastructure, and client software. They can take immediate advantage of the PKP+DNS PROVIDEs implementation to add a transparent defense from insider threats locating valuable network resources.

B. Adversarial Reaction

In our analysis we found that even using an AT of 16-bits would cause Internet scanning to be impractical for the opportunistic attacker. However the adversary is adaptive and there will always exist persistent threats. As previously mentioned, there could be an attempt to create an alternative rDNS service. To add a record, the IDT and IP address would need to be identified by either brute forcing the AT for each IP addresses or enumerating domain names and make a forward DNS query for the IDT TXT record.

A brute force approach may be taken to enumerate the fully qualified domain name space. Alternatively, they may choose a more strategic approach, for example, extracting domain names from zone files which can be obtained in bulk from the Centralized Zone Data Service (CZDS)[2] for participating Top Level Domains. Furthermore, the adversary may narrow their scope, targeting for example the one million most popular web services. Amazon offers an Alexa Web Information Service,[3] an API interacting with Alexa's data repository, which can provide domain names matching such criteria. A PKP+DNS implementation of PROVIDEs will impact next generation scanners in the following ways,

- **Increase time**: The scanner must make a DNS query to obtain the IP and IDT. The name server is a point of control, having the ability to delay and rate limit TXT queries. Additionally the IDTs are moving targets, repeated scans will need to re-query for the IDT when expired.
- **Increase complexity**: As scan times increase the task is more likely to be parallelized resulting in additional infrastructure similar to cooperative scanning techniques used by botnets [23].
- **Increase cost**: An increase in time and complexity equates to an increase in cost. The cost may be expressed as a likelihood of detection or operational costs.

Although PKP+DNS will not eliminate an advanced persistent threat (APT) from scanning by domain names, it will impose many challenges to deter an economically conscious opportunistic attacker and significantly reduce the global threat.

[2]Centralized Zone Data Service (CZDS), https://czds.icann.org/

[3]Alexa Web Information Service https://aws.amazon.com/awis/

C. Future Work

We are currently in the processes of implementing the token publisher, and token subscriber to support PKP+DNS and will deploy it in the Massachusetts Open Cloud (MOC) [24]. Our goal is to work with enterprise network administrators and other researchers to create stealth services using PKP+DNS and identify any challenges that it may introduce.

As part of this development, a performance evaluation will be conducted to determine how the clients and servers are impacted. We recognize network scanners are a valuable tool for network administrators to debug issues, and identify connected devices and running services on their network. How stealth services will affect network administration in an enterprise setting still needs to be explored.

This paper has proposed PROVIDE as a protocol to hide a server from a network scanner. However, not all network scans have malicious intent. They are commonly used by researchers as a measurement tool, yet these scans raise many ethical considerations [25]. It has been discussed in the work on ZMap [4], that it is impossible to request advance permission to scan a particular host. Currently, there is no equivalent to a `robots.txt` file to indicate the host should not be scanned [26]. We respectfully disagree. We propose that DNS PTR records be used as a signaling mechanism to allow an IP address to opt-out of being scanned. If a PTR record is set to a domain containing the label `donotscan` the scanner should skip the IP address. We have acquired the domain `donotscan.info` for others to freely use. Our intention is to work with the creators of NMap, ZMap and Masscan to integrate this signaling mechanism, and analyze the overhead it will impose on DNS.

VI. Related Work

The concept of port knocking dates back to 2001 in a Linux User Group Mailing List [27]. A method was proposed requiring a client to first access a sequence of ports before the SSH port would become visible. Three techniques were proposed by [17] to conceal services from non-authorized users. A formal security model for port knocking was proposed by [7] which is contingent on the act of port knocking remaining undetectable. Recently, an informational RFC draft for TCP Stealth has been released documenting a PKP.

Alternatively, IP hopping solutions have been proposed to defend against network scanners in enterprise networks. The threat model considers an attacker performing vulnerability scans to create a "hitlist" of victim IP addresses. These defenses frequently change the host IP addresses so they become stale by the time they are revisited by the attacker. Network Address Space Randomization (NASR) provides a solution using DHCP to update host IP addresses [28]. However, this was disruptive to active connections. OpenFlow Random Host Mutation (OF-RHM) addressed usability issues creating a transparent experience for the end user while also increasing the IP search space to be scanned [29]. Despite improvements, this defense is limited to software defined

networks (SDN). Furthermore, neither solution is scalable to Internet size networks.

DNS(SEC) has been proposed as a key distribution in many situations. Public keys have been distributed for users [30] DomainKeys Identified Mail (DKIM) [31], and opportunistic encryption [32]. Jones et al. [33] proposed the Internet Key Service (IKS) to overcome some barriers associated with key distribution in DNS. Their solutions uses DNS(SEC) to discover IKS servers which handle key queries directly rather than DNS itself.

Proofs of knowledge have been around for some time allowing a prover to convince a verifier they poses some knowledge. Zero knowledge proofs are a variant in which the prover can convince the verifier a statement is true without revealing the knowledge to do so [34], which can be used, for example, to prove an identity [35]. Authentication protocols [36], used to prove identities, include password-based, one-time passwords [15], and challenge-response authentication. Furthermore, in computer networks, the identification protocols (Indent) allows the identity of a user to be determined from a TCP connection [37].

VII. Conclusion

In response to our observation that horizontal network scanners are unaware of the targets identity, we present the PROVIDE protocol that allows a prover (client) to convince a verifier (service) they are knowledgeable of the verifier's identity. We then presented, PKP+DNS, a PROVIDE implementation using DNS for IDT distribution and a PKP for proof verification and authentication, as a method to hide from network scanners. Up until now, PKPs have been constricted to private services due to the key distribution problem. Our solution allows any service accessed by domain name, not reliant on rDNS, to be converted to a stealth service. Malicious IP-based network scans continue to be a problem, while the increase in scanning speeds, and convenience make matters worse. Fortunately, PKP+DNS has the ability to put an end to this threat. However challenges still arise in developing a unified standard for the PKP. Our objective is to work with the community in an effort to establish standards to create stealth services. We believe PKP+DNS provides a step in that direction.

Acknowledgment

This work has been supported by the National Science Foundation (NSF) awards #1430145, #1414119, and #1012798.

References

[1] "Threat feeds," http://www.dshield.org/threatfeed.html.
[2] "Meet the cryptoworm, the future of ransomware," https://threatpost.com/meet-the-cryptoworm-the-future-of-ransomware/117330/.
[3] A. Dainotti, A. King, F. Papale, A. Pescape *et al.*, "Analysis of a /0 stealth scan from a botnet," in *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM, 2012, pp. 1–14.
[4] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide scanning and its security applications," in *Proceedings of the 22nd USENIX Security Symposium*, Aug. 2013.
[5] R. D. Graham, "Masscan: Mass ip port scanner," *URL: https://github.com/robertdavidgraham/masscan*, 2014.
[6] Z. Durumeric, M. Bailey, and J. A. Halderman, "An internet-wide view of internet-wide scanning," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 65–78.
[7] E. Y. Vasserman, N. Hopper, and J. Tyra, "Silentknock: practical, provably undetectable authentication," *International Journal of Information Security*, vol. 8, no. 2, pp. 121–135, 2009.
[8] J. A. H. K. J. Kirsch, C. Grothoff, "Tcp stealth," accessed: 2016-05-17.
[9] "Port knocking implemenations," http://portknocking.org/view/implementations.
[10] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.
[11] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman, "Zippier zmap: internet-wide scanning at 10 gbps," in *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, 2014.
[12] R. Rosenbaum, "Using the domain name system to store arbitrary string attributes," 1993.
[13] M. Wong and W. Schlitt, "Sender policy framework (spf) for authorizing use of domains in e-mail, version 1," RFC 4408, april, Tech. Rep., 2006.
[14] J. Bound and Y. Rekhter, "Dynamic updates in the domain name system (dns update)," 1997.
[15] D. MRaihi, S. Machani, M. Pei, and J. Rydell, "Totp: Time-based one-time password algorithm," *Internet Request for Comments*, 2011.
[16] M. Rash, "Single packet authorization with fwknop," *login: The USENIX Magazine*, vol. 31, no. 1, pp. 63–69, 2006.
[17] P. Barham, S. Hand, R. Isaacs, P. Jardetzky, R. Mortier, and T. Roscoe, "Techniques for lightweight concealment and authentication in ip networks," *Intel Research Berkeley. July*, 2002.
[18] J. Postel, "Transmission control protocol," 1981.
[19] ——, "Internet protocol," 1981.
[20] S. E. Deering, "Internet protocol, version 6 (ipv6) specification," 1998.
[21] R. Fonseca, G. Porter, R. Katz, S. Shenker, and I. Stoica, "Ip options are not an option," 2005.
[22] "Cisco 10000 series router software configuration guide," Tech. Rep., 2010.
[23] C. C. Zou, D. Towsley, and W. Gong, "On the performance of internet worm scanning strategies," *Performance Evaluation*, vol. 63, no. 7, pp. 700–723, 2006.
[24] "Massachusetts open cloud," http://info.massopencloud.org/.
[25] S. Jamieson, "The ethics and legality of port scanning," *SANS Institute*, 2001.
[26] M. Koster, *A standard for robot exclusion*. NEXOR., 1994.
[27] C. Borss, "Drop/deny vs. reject," *Listserv post to Braunschweiger Linux User Group (lug-bs@ lk. etc. tu-bs. de). Available at: http://web.archive.org/web/20060618092902/http://www.lk.etc.tu-bs.de/lists/archiv/lug-bs/2001/msg05734.html*, 2001.
[28] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis, "Defending against hitlist worms using network address space randomization," *Computer Networks*, vol. 51, no. 12, pp. 3471–3490, 2007.
[29] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 127–132.
[30] J. M. Galvin, "Public key distribution with secure dns." in *USENIX Security*, 1996.
[31] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas, "Domainkeys identified mail (dkim) signatures," RFC 4871, May, Tech. Rep., 2007.
[32] M. Richardson and D. Redelmeier, "Opportunistic encryption using the internet key exchange (ike)," Tech. Rep., 2005.
[33] J. P. Jones, D. F. Berger, and C. V. Ravishankar, "Layering public key distribution over secure dns using authenticated delegation," in *Computer Security Applications Conference, 21st Annual*. IEEE, 2005, pp. 10–pp.
[34] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on computing*, vol. 18, no. 1, pp. 186–208, 1989.
[35] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of cryptology*, vol. 1, no. 2, pp. 77–94, 1988.
[36] J. Clark and J. Jacob, "A survey of authentication protocol literature: Version 1.0," 1997.
[37] M. S. Johns, "Identification protocol," 1993.