

6.857 Final Project: Cloaked Server Security

Robert Beverly
rbeverly@mit.edu

Richard Hansen
rhansen@mit.edu

Vijay Shah
vjshah@mit.edu

ABSTRACT

The ability to secure systems that provide non-public services over public network infrastructures such as the Internet is challenging. Even when these services use cryptographically secure authentication, implementation bugs are quickly exploited by attackers to turn these services into gaping security holes. We introduce the notion of “Cloaked Server Security,” in which authorized users are able to easily connect to the desired service, but unauthorized users cannot determine if the server exists. We formalize this concept by using an iterative approach to carefully define the model for our adversary and the challenge presented to that adversary. In evaluating a number of common and interesting network security schemes against our definition, we find most of them lacking against even weak adversaries. However, anonymizing networks provide the most promising approach to meeting our security definition.

1. PROBLEM STATEMENT

This paper presents and formalizes a new network security definition: “Cloaked Server Security”. Rather than producing a new security scheme, we focus on precisely defining an appropriate notion of network security. In much the same way that Naor and Yung formalize Chosen Ciphertext Attacks (CCA) in the context of public key cryptosystems [15] and Goldwasser, Micali and Rivest define Adaptive Chosen-Message attacks against digital signature schemes [9], we posit that it is impossible to produce acceptable security implementations before defining the security model adequately. Cloaked Server Security is intended to serve as a metric against which to evaluate network security proposals.

The motivation for our work comes from several practical ramifications of today’s highly interconnected Internet—with no shortage of malicious adversaries. First, network services perpetually suffer from security flaws and bugs in their implementation. That is, there is a well-known gap between the theoretical security of a service and the actual security of a particular implementation. As an example, the OpenSSL project [1] distributes a library of cryptographic routines that provide a secure socket layer abstraction. Many popular applications, e.g. Secure Shell (ssh), Apache Web Server with mod_ssl, Secure Email (IMAPS), etc. use this library as part of their service¹. A casual pe-

¹The problems that can arise when multiple services share a single flawed library are beyond the scope of this work. However, we note that such a simultaneous failure mode is interesting from a security perspective.

rusal of security advisory databases reveals numerous OpenSSL implementation flaws, even as recently as one year ago [4, 13].

Similarly, the Slammer worm [14] exploits an implementation problem in Microsoft SQL server and demonstrates a growing trend toward automated attacks. These and many other security vulnerabilities are not obscure, isolated incidents, but rather common and with wide-reaching implications. In addition, they are actively and continually exploited by both individual hackers as well as automated worms and bot-farms. Cloaked Server Security is particularly well suited against this class of automated adversaries.

Second, network machines may leak information valuable to an attacker. The popular and readily available `nmap` utility [7] can remotely scan hosts or entire networks to determine which services are running. Using inference on TCP stack behavior, the scan can even reveal a machine’s uptime and operating system type and version. In this way, scanning often provides a wealth of information to even a very weak adversary, one who has no control over her location in the network and has not yet infiltrated the remote machine.

Knowing potential attack points saves an adversary time and effort. Additional information improves the efficiency of automated worms and attacks, effectively raising the attacker’s compromised machine yield. For these reasons, our thesis is that network servers *should not leak any information, even their very existence*. Informally, we define “Cloaked Server Security” as any solution that hides the existence of a publicly-connected server to all but authorized users.

2. POTENTIAL SCHEMES FOR CLOAKED SERVER SECURITY

This section briefly reviews four existing schemes that provide an approximation of Cloaked Server Security. Throughout this section we refer to an adversary or attacker in generic terms purely for expository purposes. Our taxonomy in Section 3 concretely defines the adversary and evaluates each of the systems presented here against our definition of Cloaked Server Security. Our evaluation of real-world systems gives our work an additional measure of practical applicability.

- **Firewall:** The term “firewall” generically refers to any system that attempts to police the traffic between two

network segments by permitting or rejecting data according to some security policy [5]. There are many different techniques used to implement policies; one of the more common methods for IP networks is to create a ruleset based on a tuple of the IP packet's source and destination addresses and ports. This method is known as a packet-filtering gateway firewall. When referring to firewalls in this paper, we assume a packet-filtering gateway. An underlying assumption for firewalls is that a packet's relevant IP header information is accurate. However, recent research has shown that spoofing IP header information, including the source address, is often possible even on today's Internet [2]. We incorporate this fact into our model of the adversary.

- Virtual Private Network:** Virtual private networks (VPNs) attempt to emulate a private network over public infrastructure, primarily for economic advantage. A private network, where individual wires are physically run to each individual authorized participant in the network, provides security from non-physical outside attack (but not from internal attack). Because of the often unrealistic expense and inflexibility of a private network, there are several means to provide a virtual approximation. There are two general types of VPNs: trusted and secure [8]. Trusted VPNs rely on a provider's network to protect the traffic, typically using some encapsulation mechanism. In contrast, secure VPNs use cryptographic tunneling. Our interest lies with secure VPN schemes such as IPsec [11]. IPsec uses a combination of cryptographic authentication and encryption to secure an untrusted communication path between two gateways.
 - Port Knocking:** Port knocking is a relatively recent technique [12] developed as a means towards making servers seem invisible. An adversary running a scanning utility against a system secured with port knocking is shown in Figure 1. From the attacker's perspective, a server using a port knocking-based defense appears to silently discard all incoming packets, regardless of the connecting peer's IP address or port. In this way, the attacker receives no responses and effectively cannot discern whether the server exists. However, authorized clients have a way to modify this behavior. Each authorized client knows a secret "knock," which is simply a secret encoded as a series of numbers drawn from the domain of possible TCP ports (1-65535). When the client wishes to access a service on a protected machine, she initiates TCP connections to each of the ports specified by the secret knock in succession. Although the server drops each of these connection attempts, it is silently noting the sequence of ports accessed by each remote IP address. When the correct sequence is completed, the server changes its behavior such that access to the service is now granted to the client's IP address for a short amount of time. Figure 2 shows an authorized user accessing a service using port knocking. Port knocking can therefore be thought of as a remotely configurable firewall.
- A major flaw in this scheme as described is that it is vulnerable to replay attacks. An adversary who is able to observe the knock sequence (by compromising the

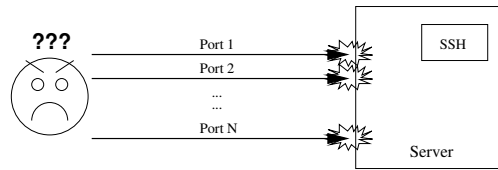


Figure 1: Port knocking: the server silently discards all of the adversary's probes. The Adversary cannot determine if server even exists.

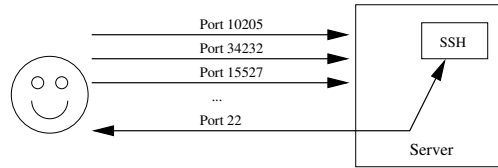


Figure 2: Port knocking: authorized users send a series of "knocks" by initiating unsuccessful TCP connections. With the correct knock sequence, the service is available to the knocker.

client machine, or any of the other intermediate machines on the route to the server) can later transmit that same sequence to gain unauthorized access. Various improvements to the basic scheme have been suggested to correct this flaw, such as the use of one-time knocks [3], or strongly tying the client to the knock with encryption using the IP address as the key [12]. However, these improvements do not significantly improve the security of port knocking under our security definition, and therefore it suffices to assume the most basic version as an abstraction of the general scheme.

- Anonymizing Network:** TCP/IP provides no degree of anonymity; a packet's source and destination IP addresses are plainly visible to both peers of a session and to any machines along the connecting path. Determining the real-world identity that corresponds to a given IP address at a given time is usually not difficult, especially with the help of external agents, such as the court system. There are many reasons why people would like to ensure privacy when communicating over the Internet, on both the client and server sides. For example, an organization operating under an oppressive regime might want to offer information that the government considers subversive, without exposing any identifying information that would allow the government to shut it down (or worse). Likewise, citizens living under this government want to retrieve this information without the possibility of being traced. Anonymizing networks create a virtual network overlay on top of TCP/IP that allow parties to use existing networks and protocols (such as HTTP over the Internet) to communicate without exposing any sensitive personal information.

One such network that is in actual production today is Tor [6]. Tor uses onion routing to provide anonymity; clients on the network have the ability to construct a path to any other peer such that all intermediate

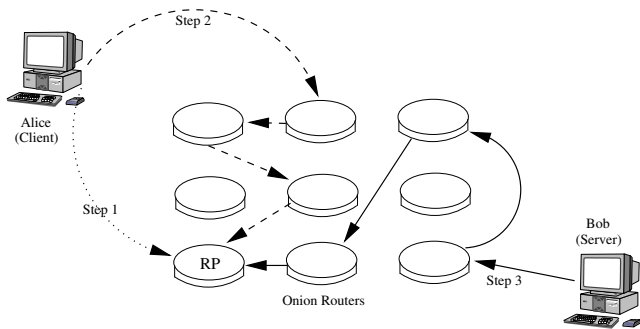


Figure 3: Anonymizing Network: 1) Alice randomly selects a rendezvous point (RP) from the set of onion routers (ORs) and asks Bob to meet her at the RP. 2) Alice constructs a random path to the RP through the ORs. Any single OR knows only the ingress and egress neighbor ORs for a traffic stream, not the complete path. 3) The server similarly constructs a random path through the ORs to the RP. Communication is now possible without Alice or Bob knowing each other’s origin.

routing points only know the IP address of the ingress and egress neighbors. Additionally, any intermediate point cannot tell how many other routers are between it and the connecting peers.

While Tor is primarily intended to provide anonymity, we are particularly interested in the security properties offered by Tor’s concept of “hidden services.” A server that wants to establish a service first chooses a small set of other Tor nodes at random. These nodes will be the service’s “introduction points.” The server encrypts a list of the introduction points using a symmetric key k , and publishes the pair (hostname, E_k (introduction point list)) in some globally available table. Clients authorized to use this service will be given the hostname and symmetric encryption key k out of band.

When a client wants to connect to the service, she first randomly chooses a node to serve as a “rendezvous point” (RP). She then constructs a random path through the network of onion routers to the RP. After she fetches and decrypts the list of introduction points for the service, she connects to one of these introduction points and transmits her name, selected rendezvous point and a random cookie, all encrypted with the server’s public key. In addition, the message to the introduction point is signed with the client’s private key, allowing the server to authenticate the client. The introduction point relays this message to the server. The server decrypts the message and connects to the RP, using a randomly signaled path, sending it the random cookie. At this point, 2-way authentication is complete; both the client and the server are assured they are communicating with the intended party. They use the RP to relay all subsequent data. The anonymizing network communication setup process is summarized in Figure 3. Note that neither the client nor the server knows the other peer’s address, nor does the rendezvous point know the address for either party.

When we refer to anonymizing networks in this paper, we are not specifically discussing Tor, although Tor is the only network we are aware of that offers hidden services. We are interested in these high-level properties for anonymous client-server communication: the client and server should be able to find each other and communicate without knowing each other’s location, and no one else on the network should be able to discover either party’s identity or location.

3. BUILDING AN APPROPRIATE SECURITY TAXONOMY

Algorithms have precise mathematical structure that can be exploited to make absolute claims about security. Systems, on the other hand, resist absolute claims due to the complexity and variability of the real world. It is possible to create mathematical models that approximate real behavior, but the process is quite difficult. Without creating a mathematical model, defining the security of a system is an imprecise business; there are just too many uncertainties. Yet we attempt to do just that in this section, acknowledging up front that the definitions we specify are neither complete for any particular system nor do they lend themselves to absolute mathematical security proofs. However, we feel that this work does provide real value by providing criteria for evaluating the quality of defensive countermeasures given realistic threats.

Our approach to defining Cloaked Server Security is to speculate on the fundamental abilities of real-world adversaries. Along the continuum of potential adversaries, we attempt to choose an attacker that is sufficiently strong to provide realistic security without being so strong that no scheme can ever suffice. We build our security definition iteratively by evaluating each candidate against existing defensive schemes, ending with the strongest model. Our last, strongest security model is most inclusive of the broad spectrum of potential adversaries seen in common Internet attacks.

3.1 Correct Behavior

Before presenting details of incorrect behavior where the security of the system is violated, it is important to acknowledge *proper* behavior: Authorized clients are able to deliver packets to and receive packets from the cloaked server on demand. It does not suffice to merely send a packet to the server, the service the packet is intended for must receive the packet. In practice, this communication will often correspond to a two-way session. Communication with the server be direct or indirect, but must be reliable relative to the underlying transmission network. In addition, clients are permitted to perform a setup phase before establishing communication. There are no constraints on the the packet payload; it may contain arbitrary information.

3.2 Lone Adversary

Security Definition

The ability of an adversary to subvert remote access control mechanisms centers around her ability to monitor and manipulate network traffic for a given server and/or its authorized clients. The location of the adversary has a great effect on this ability, as she has no access to the data stream

for a given connection if she is not located on the same subnet² as one of the peers, or at some intermediate point on the route between the peers. An attacker who is located at one of these sensitive points is very powerful; for our first and weakest adversary definition, we will assume that the attacker is located elsewhere. However, the attacker can attempt to take over other machines that *are* at sensitive locations.

The types of take-over attacks on network-connected machines can be separated into two classes: *opportunistic* take-overs, where the attacker has specific a vulnerability (or set of vulnerabilities) she targets for exploit, and searches for machines that exhibit that vulnerability; and *targeted* take-overs, where the attacker has a specific machine that she wants to compromise, and searches for an attack vector against that machine. Because of the vast size, diversity and connectedness of the Internet, opportunistic take-overs are much easier to execute than targeted take-overs. We define our *Lone Adversary* as an individual who is able to perform limited opportunistic take-overs, but not targeted take-overs. We assume that this adversary wants to minimize her risk, and therefore will only try to compromise a few machines (< 10). It is important to note that because the adversary has no control over which machines exhibit the vulnerability she is exploiting, she has no control over where her victims will be located relative to her target. While we acknowledge that this adversary is very weak, we believe she represents a large class of potential attackers on today's Internet³.

We must define the degree to which the Lone Adversary can manipulate network traffic. Native TCP/IP does not cryptographically ensure the authenticity of its header information, so one popular attack is to either modify existing packets or create new packets such that they have forged source location information. This attack is known as *IP spoofing*. An attacker can use IP spoofing to establish a new connection that appears to come from a falsified origin, disrupt an existing connection, or transparently take control of an existing connection (*session hijacking*). This is a powerful attack vector, but also one that has become somewhat difficult to employ on today's Internet. Ingress and egress packet filtering rules in routers make it harder for an adversary to successfully change the header information, and security features in modern TCP/IP stacks, such as unpredictable TCP sequence numbers, make it harder to establish a spoofed TCP connection. Our basic Lone Adversary will not have the ability to spoof IP header information. She is, however, free to modify packet payloads.

Given this adversary, we now define the challenge she must

²The term "subnet" refers to a portion of the network under common administrative control where all hosts on the subnet share a common address prefix. We present subnets here because they are an important and realistic consequence of being connected to the Internet. It is often the case that an adversary on the same subnet as her victim or her victim's traffic has a security advantage.

³The number of machines the adversary takes over may also be limited by motivation, resources or ability. We are not artificially weakening the adversary, but rather modeling a realistic class of attacker. We strengthen the adversary significantly in the next subsections

meet in order to breach Cloaked Server Security. To provide a strong degree of security, the challenge is fairly easy to meet: the adversary succeeds if she can deliver a single packet to an active service on the server. In other words, the system or network must guarantee that any packets originating from or manipulated by unauthorized users will be dropped before they reach the application layer. As the Slammer incident demonstrated, a single packet can be all that is required to compromise a system.

Evaluation of Security Schemes

- **Firewall:** The packet filtering firewall denies all traffic with unauthorized source IP addresses. Because we have defined our adversary such that she cannot forge her source information, in order to bypass the firewall she must have under her control a machine in a sensitive location. The sensitive location is either a machine with an IP address authorized in the firewall's ruleset or a machine along the route between an authorized client and the server. As argued above, the adversary has no control over the location, and hence addresses, of her compromised machines. For a network of size α , the existence of β sensitive locations on the network, and a set of γ compromised machines drawn uniformly from the network, then the adversary will succeed in compromising a sensitive location with probability

$$1 - \left(\prod_{k=0}^{\gamma-1} \frac{\alpha - \beta - k}{\alpha - k} \right) \quad (1)$$

Taking $\beta = 1K$, $\gamma = 10$ and using a realistic value for the number of hosts on the Internet, $\alpha = 500M$, the adversary succeeds with a rough probability of 2×10^{-5} . While this probability is clearly not cryptographically strong, it still provides a reasonable measure of practical security. Specifically, any adversary that is strong enough to mount a coordinated attack and significantly improve her odds falls into our next class of adversary.

- **Port Knocking:** The analysis for the port knocking scheme is very similar to the analysis for Firewalls as given above. In order to bypass the port knocking access control mechanism, she must have under her control a machine along the route from a client who authorizes himself to the server. Over time, the value β will be on the same order as the value for the case of firewalls. Therefore, we claim that port knocking provides a similar degree of protection against this adversary.
- **VPNs:** This defensive scheme is secure for a lone adversary. Even if the adversary happened to locate herself between the client and the server, she could not successfully insert a packet into the cryptographically protected tunnel. The secure authentication prevents the attacker from establishing her own tunnel to the gateway. Therefore, she is unable to get a packet to the server in the first place, let alone a higher-layer process.
- **Anonymizing Networks:** This defensive scheme is also secure with a lone adversary. The attacker does

not know the location of the server and can not determine it based on traffic exchanged amongst the anonymizing routers. Therefore, it is not possible for the attacker to send a packet to the server.

3.3 Botnet Adversary

Security Definition

Because opportunistic take-overs are easy, they are also very fruitful; an individual who deploys a worm designed to automatically perform take-overs can quickly assume control of an army of tens of thousands of machines. These armies are popularly known as “botnets.”

The second adversary we define is called the *Botnet Adversary* because she has the ability (and the willingness to accept the risk) to perform these automated opportunistic take-overs. We still restrict her from performing the harder targeted take-overs. In addition, the Botnet adversary has the ability to spoof, or forge, the identity of packets in the network.

As in the previous section, the adversary has no control over which machines have the vulnerability she’s searching for, so she cannot control the location of her soldier machines. She must hope that the size of her army is large enough to probabilistically ensure that she ends up controlling a machine in a sensitive location.

We present the same challenge to our Botnet Adversary that we gave to the Lone Adversary; she succeeds if she can send a single packet to an active service on the server.

Evaluation of Security Schemes

- **Firewall:** The firewall aims to deny all traffic with unauthorized source IP addresses. Because the adversary is given the ability to spoof, the security of firewalls immediately fails. A remote attacker can simply assume the identity of a trusted host to subvert the firewall. However, even in a world where spoofing is not feasible, the security of firewalls still fails by a probabilistic argument. In order to bypass the firewall without spoofing, the attacker must have under her control either a machine with an authorized IP address or a machine along the route between an authorized client and the server. As argued above, the adversary has no control over the location of her soldier machines. Reconsidering equation (1) with $\gamma = 100K$, a realistic figure for today’s large botnets, attackers have approximately a 20% probability of success. This surprising result highlights the power of botnets.
- **Port Knocking:** The adversary’s ability to spoof packets immediately defeats the security of port knocking. Because there is no strong coupling between the knocking process and a session, an adversary can hijack a session through spoofing. The attacker waits for an authorized party to open the service and then connects while the door is open before the true client can. However, as with firewalls, even in a situation where spoofing is not possible, the sheer size of the botnet gives the adversary a non-negligible chance of compromising a machine that can passively view the

knock sequence enabling a replay attack.

- **VPNs:** Because of the strong cryptographic two-way authentication of VPNs, they remain secure against the Botnet adversary.
- **Anonymizing Networks:** The anonymous nature of services prevents an adversary from even knowing if a compromised machine is along the routing path of a circuit in an anonymizing network. Thus, the server remains effectively hidden from a botnet attack.

3.4 Mobile Adversary

Recognizing that our security definitions thus far are somewhat weak in light of the range of Internet adversaries and attacks, we present our final, strongest definition of Cloaked Server Security. We strengthen the security definition in two ways: giving the adversary more power and relaxing what it means to compromise system security. We then evaluate the four security solutions under this new definition.

Security Definition

To strengthen the adversary, we follow the notion of a *mobile adversary* as defined in [10]. This adversary has the targeted take-over ability discussed earlier: she can place herself anywhere in the network. This power assumes that the adversary can somehow communicate with her target, which implies that she knows where the target is and can deliver packets to that location. The adversary can place herself at multiple locations, but, due to the effort required to compromise a location, we limit the take-over rate.

As before, the adversary has complete power over the network traffic at her location. She can observe, delete, and insert arbitrary packets. By combining these abilities, she can modify, delay, or exchange the order of packets.

However, the adversary’s unwelcome presence is detectable. Once detected, she can be removed. We assume that security mechanisms are in place everywhere to eventually detect and remove an intruder. This, combined with the limited take-over rate, places an upper bound on the number of locations the adversary can occupy at any given time.

We acknowledge that a mobile adversary is very powerful. However, this adversary is appropriate given the current state of the Internet; motivated and well-funded adversaries have this power. Furthermore, by varying the take-over and removal rates, our model is inclusive of a range of realistic scenarios. Real world adversaries, those of varying degrees of motivation and support, and real world targets, with different degrees of defensive abilities, are accounted for by the two parameters to our adversarial model: how fast the adversary can take over hosts and how quickly she is detected.

We can view the temporal aspect of our adversary as a queuing problem. Formally, let λ be the attacker’s rate of newly compromised hosts and μ the detection and removal rate. For simplicity, assume that take-overs and removals follow a random exponential (Markovian) distribution. μ must be greater than λ or the number of compromised machines would grow infinitely. The adversary’s pool of available machines can be thought of as corresponding to the number of machines in a M/M/1 queue. The utilization of the queue

is $\rho = \frac{\lambda}{\mu}$ which we use to compute the average number of compromised machines available to the attacker at any given moment:

$$\hat{N} = \frac{\rho}{1 - \rho} = \frac{\lambda}{\mu - \lambda} \quad (2)$$

This number can help determine the security of a scheme that relies on the proper behavior of a collection of multiple systems to provide security to the server.

With a strengthened adversary, we now discuss relaxing what it means to compromise security. Our weaker adversaries defined a security breach as an event where a packet is successfully delivered to a higher-layer process. In Mobile Adversary model, merely delivering a packet to the machine, directly or indirectly, is enough to compromise the server. The justification for this seemingly-impossible-to-prevent event is the prevalence of resource depleting Denial of Service (DoS) attacks. These attacks do not require packet delivery to the higher-layer process to cause damage; merely getting the packet to the server consumes critical resources.

In addition to the above packet delivery requirement, we require that an adversary cannot determine whether the server even exists. In other words, if the attacker can tell the difference between a server and an empty network jack, then security is compromised.

Formally: Let $A()$ be the adversary, M be the network address for a server, N be an address that is not associated with a machine, and $\delta()$ be a negligible function in the size of the network:

$$\begin{aligned} r &\stackrel{R}{\leftarrow} \{0, 1\}^1 \\ \alpha &\leftarrow (rM + (1 - r)N) \\ \beta &\leftarrow (rN + (1 - r)M) \\ d &\leftarrow A(\alpha, \beta) \\ \Pr(d = M) &\leq 1/2 + \delta() \end{aligned}$$

The above two security requirements (unknown existence and inability to send packets) overlap somewhat, but they are not redundant. Even if an attacker does not know whether the server exists, she can still hypothesize and lob packets in the hope of potentially disrupting the server or the server’s network.

A summary of our strongest definition of Cloaked Server Security is given in Figure 4.

Evaluation of Security Schemes

- **Firewall:** This scheme remains insecure. By our definition, the adversary can simply target the firewall, take it over, and control all of the traffic passing through.
- **Virtual Private Network:** By the same argument as above, this scheme is now insecure. The adversary simply targets the VPN gateway. Once the VPN gateway is taken over, the adversary has complete access to decrypted traffic entering the server’s local network. The adversary can determine the existence of the target server by examining this traffic, thus winning the

ADVERSARY’S ABILITIES

- Adversary can choose *any* single point (host, link, router) to put herself
- Adversary can slowly take over other choice points at rate λ
- Adversary is eventually detected and removed from these points at rate μ
- Adversary can observe packets, delete packets, and insert new (possibly spoofed) packets

SECURITY REQUIREMENTS

- Security is compromised if an adversary discovers that the target server exists
- Security is compromised if an adversary can deliver a packet to the server

Figure 4: Cloaked Server Security definition summary (mobile adversary)

game. Or, she can trivially insert new packets.

- **Port Knocking:** This scheme remains insecure. The adversary can directly compromise the target server, or she can target a nearby host to observe the knock.
- **Anonymizing Network:** This scheme remains secure. Because the identity of communicating peers remains unknown to everyone except the peers, the adversary is unable to determine what traffic might be destined for the target server. Therefore, she is unable to determine the existence of the server.

Even if she were able to identify traffic destined for the target server, tracing the traffic to the server’s location would require compromising all of the routers leading back to the server. This can be made arbitrarily difficult by increasing how often the server’s random path changes relative to the attacker’s take-over rate. Therefore, the task of delivering a packet to the server can be made arbitrarily difficult.

In essence, on an anonymous network, no host has a discernible location and all traffic looks like pure noise. Therefore, no information can be gleaned and the server remains protected.

4. CONCLUSIONS

To the best of our knowledge, our work is the first attempt to formally codify the notion of cloaked security. Table 1 summarizes our findings. Although, by necessity, we cannot be completely mathematically rigorous about our definition and proofs of security, we feel that we have been careful enough in our definitions that they can serve as useful criteria by which one can confidently reason about specific implementations of cloaked security.

We join the chorus of security professionals proclaiming that firewalls must not be used as the lone defense measure for an organization. Not only do firewalls not provide cloaked security, but as we have shown mathematically, they perform poorly even at the job they are designed for: limiting access

Table 1: Cloaked Server Security Matrix: Summary of scheme security versus adversary strength

Scheme	Adversary		
	Lone	Botnet	Mobile
Firewall	✓		
VPN	✓	✓	
Port Knocking	✓		
Anonymizing Network	✓	✓	✓

to a network segment to a set of authorized IP addresses. The advent of botnets makes firewalls an especially weak tool. An adversary with a large botnet has roughly a 20% chance at compromising the average organization!

We find it interesting that the concept of port knocking inspired our initial desire to define cloaked security, and yet could only meet the weakest of our definitions (and even then, only with reservations). In the end, port knocking fared no better than firewalls. This failure underscores the point made in our introduction: defining a security model is a necessary first step to creating a security implementation.

VPNs meet a weakened, but still useful, definition of cloaked security while maintaining efficient network operation. However, system administrators must ensure that the VPN is only part of a larger network security strategy, taking pains to avoid creating a so-called “hard outside, soft center” model. No security system should rely on the soundness of a single machine.

The success of anonymizing networks in meeting our strongest definition of cloaked security was a pleasant surprise. We must note that current implementations of anonymizing networks are not without their drawbacks; bandwidth and latency are both much worse over the anonymous overlay than over TCP/IP, making these networks too slow for most purposes at the moment.

We believe that the concept of cloaked security is relevant for many organizations and should be an important feature of future network architectures. We hope that our findings will help spur the participation and research necessary to make anonymizing networks a widespread practical reality.

5. REFERENCES

- [1] OpenSSL. <http://www.openssl.org/>.
- [2] R. Beverly and S. Bauer. The spoofer project: Inferring the extent of source address filtering on the internet. In *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI) Workshop*, pages 53–59, 2005.
- [3] Capella and T. C. Keong. Remote server management with one-time port knocking (OTPK). June 2004.
- [4] CERT. Multiple vulnerabilities in SSL/TLS implementations, CA-2003-26, 2003. <http://www.cert.org/advisories/CA-2003-26.html>.
- [5] W. Cheswick and S. Bellovin. *Firewalls and Internet Security*. Addison-Wesley, 1994.
- [6] R. Dingedine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
- [7] Fyodor. Remote OS detection via TCP/IP stack fingerprinting, 1998. <http://www.insecure.org/nmap>.
- [8] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A Framework for IP Based Virtual Private Networks. RFC 2764 (Informational), Feb. 2000.
- [9] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
- [10] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *ACM Conference on Computer and Communications Security*, pages 100–110, 1997.
- [11] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401 (Proposed Standard), Nov. 1998. Updated by RFC 3168.
- [12] M. Krzywinski. Port knocking: Network authentication across closed ports. 12:12–17, 2003.
- [13] MITRE. Common vulnerabilities and exposures, CVE-2004-0112, 2004. <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0112>.
- [14] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. 1(4):33–39, 2003.
- [15] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *ACM Symposium on Theory of Computing*, pages 427–437, 1990.