Network Security

Pamela Zave and Jennifer Rexford

December 2, 2018

Network security can be divided into two major categories, based on where it is provided. *Endpoint security* consists of security measures implemented primarily in endpoints that wish to communicate, and do not trust the rest of the network between them. Endpoint security is always implemented with cryptography. *Infrastructure security* consists of security measures implemented primarily by infrastructure members on behalf of the network's administrative authority, so that the network can provide its specified services. Infrastructure security is usually implemented with packet filtering. The issue of *privacy* is closely related to infrastructure security, because it is often concerned with limiting the power of the infrastructure.

In principle, any security measure might be found in any network in a compositional architecture. In addition to explaining the basics of network security, we will consider how security mechanisms interact with other mechanisms within their networks and across composed networks. Our goal is to understand where security could and should be placed in a compositional network architecture.

1 Security provided by endpoints

Currently in a separate document.

2 Security provided by infrastructure

The administrative authority (AA) of a network is responsible for protecting infrastructure members and well-behaved user members from attackers. It is also responsible for providing network services as specified. The infrastructure members are controlled by the AA and trusted by it to perform security tasks with this goal.

This section begins with an overview of the diverse goals of infrastructure security (§2.1). Regardless of the goal, most infrastructure security is provided by some kind of packet filtering, so in the bulk of the section §2.2 covers packet filtering and §2.3 covers security mechanisms that are not packet filtering. In §2.4 we return to the subject of composition, considering the interactions among security mechanisms, composition operators, and other aspects of networking.

2.1 Goals of infrastructure security

For infrastructure security, the goals are complex, as they are contingent on threat models as well as many assumptions and interests. This section divides goals into two major types.

2.1.1 Preventing or mitigating resource attacks

A *resource attack* seeks to make its victim unavailable by exhausting its resources. In networking, resource attacks are usually called *flooding attacks,* because they entail sending floods of packets toward the victim. Flooding attacks are one type of *denial of service (DoS) attack*.

The intended victims of flooding attacks vary. If the victim is a public server or other endpoint, the attack might seek to exhaust its compute-cycle or memory resources. An attacker might also target some portion of a network, seeking to exhaust the bandwidth of its links. A bandwidth attack can make particular endpoints unreachable, and can also deny network service to many other users whose packets pass through the congested portion of the network. Note that some public servers such as DNS servers are part of the infrastructure of a network, so a flooding attack on a DNS server is an attempt to deny network service to a large number of users.

If an attacker simply sends as many packets as it can toward a victim, the resources expended by the attacker may be similar to the resources expended by the victim! For this reason, an effective flooding attack always employs some form of *amplification,* in which the attacker's resources are amplified to cause the victim to expend far more resources. Here are some well-known forms of amplification:

- A "botnet" (see sidebar) is formed by penetrating large numbers (as in millions) of innocent-but-buggy Internet members, and installing in them a particular kind of malware. Subsequently the attacker sends a triggering packet to each member of the botnet, causing it to launch a security attack unbeknownst to the machine's owner. A flooding attack from many network members, particularly members of a botnet, is called a "distributed DoS attack."

- An "asymmetric attack" sends requests to a server that require it to expend significant compute or storage resources for each request, so that a relatively small amount of traffic is sufficient to launch a significant attack. A typical IP example is a "SYN flood," in which the victim receives a flood of TCP SYN packets. Each packet causes the server to do significant work and allocate significant resources such as buffer space. Also in IP networks, attackers can flood DNS servers with random queries (a "random subdomain attack"). These will force the servers to make many more queries, because they will have no cached results to match them. In a Web-based application network, the attacker can send particular HTTP requests that force the Web server to do a large amount of computation.

- An attacker can send many request packets to public servers, with the intended victim's name as source name. This "reflection attack" causes all the servers to send their responses to the victim. It amplifies work because responses (received by the victim) are typically much longer than requests (sent by the attacker).

- In an Ethernet network, a forwarder's response to receiving a packet to an unknown name is to broadcast it across the network. An attacker can amplify any packet by broadcast, simply by putting in an unused destination name.

If network infrastructure discovers where attack traffic is coming from, it can often block traffic from the attacker to stop the attack (or, eventually, take legal action). For this reason, attackers employ various techniques to hide themselves, for example:

- In an IP network, a sender can simply put a false source name in the packet header. This is necessary for reflection attacks. It cannot be used if the attack entails a dialogue with the victim, because a false ("spoofed") source name would prevent the dialogue.

- With a botnet, none of the bots sending attack traffic are actually responsible for the attack. Even if bots use true source names, there may be too many of them to cut off.

- An attacker can hide by putting a smaller-than-usual number in IP packets' "time-to-live" fields, so that the packets disappear after they have done their damage in congesting the network, but before they reach a place where defenses are deployed.

The examples of amplifications and hiding techniques show that flooding attacks are network-dependent, because they exploit vulnerabilities in the protocols of specific networks. Nevertheless, their effects are *not* network-dependent, because of "fate sharing." All the network members and applications on a machine share the same physical resources and physical network link, so if resource exhaustion causes a machine to crash, thrash, or become disconnected, all programs running on the machine will share the same fate.

Flooding attacks are a very serious problem in today's Internet. There are businesses that generate them for small fees. They target popular Web sites and (especially) DNS [9]. The worst attacks are mounted by enterprises, albeit illegal ones, that can draw on the same kind of professional knowledge, human resources, and computer resources that legitimate businesses and governments have. Such attackers will use many attacks and combinations of attacks at once, and can continue them over a long period of time.

2.1.2 Blocking specific communications

Obviously, the default behavior of a network is to provide all communication services requested of it. These services should be provided according to explicit or

implicit agreements about quality, privacy, and billing. There are, however, specific communications that a network treats differently and prejudicially. These communications are prevented, secretly recorded, or tampered with in some other way.

Here are some well-known examples of specific communications that may be prevented using the mechanisms of infrastructure security:

- Email spam and voice-over-IP robocalls should not be delivered.
- Malware should not be delivered.
- Two endpoints can willingly participate in illegal communication. This should be prevented, or in some cases recorded for further investigation or evidence in legal proceedings (the industry term for this is "lawful intercept").
- Two endpoints can willingly participate in communication that violates parental controls, which should be prevented.
- Operators of enterprise networks know which employees are using which machines for which purposes. Often they configure their networks to prevent unnecessary communication, which is probably a mistake and may be an attack. For example, machines used by engineers should not have access to the enterprise's personnel database.
- Port scanning is the process of trying every TCP or UDP destination port on an IP endpoint, to see if it will accept a session initiation. Port scanning does not in itself do much harm, but should be prevented because it is gathering information to be used in launching other security attacks. (Most malware targets a known vulnerability in a specific program or application.)

Malware is particularly dangerous when it attacks the control mechanisms of a network, such as its directory and routing protocol, or basic utilities such as clock synchronization and certificates. These can be straightforward DoS attacks, or something more insidious. Attacks on directories and routing can be parts of other attacks, because they are such fundamental building blocks that other defenses rely on them. By subverting directory entries or forwarding tables, attackers can draw packets with other destinations to themselves. Having done this, the attackers can read, absorb, inject, or alter packets as they are transmitted (these are the threats to transmission enumerated at the beginning of §1). Attackers can also impersonate the intended destination, thus stealing commerce or secrets.

2.1.3 Protecting freedom and privacy

Endpoint security keeps the content of network communications secret, but it does not hide the fact that the endpoints communicated. Even with encryption, observables such as packet headers, packet size, and packet timing yield

plenty of information. These packet attributes are observable by the network infrastructure as a matter of course, and may also be observable by third parties who tap a wire, put a wireless receiver near a wireless transmitter, connect to a wired broadcast medium such as as an Ethernet or cable network, or penetrate an infrastructure machine. "Packet sniffing" software is readily available to help them do it.

Network infrastructure can use the observable information to monitor and censor the network activities of users. Endpoints such as Web servers can use the information to keep track of who is accessing the servers. Third-party snoopers can use it for personal or commercial surveillance.

All of these uses can compromise the freedom and privacy that network users have a right to, from a legitimate or ethical viewpoint. What some national governments consider law enforcement, others consider retaliation against political dissidents. So there is value in building technology to help users evade censorship and protect their privacy. But it is very important to note that these are social and legal, rather than technical, distinctions. The constraints new technology is seeking to evade may be exactly the same constraints that technology was seeking to enforce in §2.1.2. The best technology, in keeping with the "tussle" philosophy of [6], is technology that accommodates all possible outcomes of social, legal, and commercial debates.

Personal data privacy is a related issue that is much more widely discussed. People today are concerned about the massive amounts of personal data that is collected about them by Web sites and other applications. This data is extremely valuable for selling advertising, and can also be used for worse purposes. Network privacy—privacy about one's usage of a network—can contribute to personal data privacy, but only in a limited way. For example, people can access search engines and read Web sites anonymously, at the cost of longer delays and worse search results (because they are not customized). On the other hand, people cannot participate in social media or electronic commerce in any meaningful way while preserving anonymity.

2.2 Security by packet filtering

Packet filtering is by far the most important mechanism for infrastructure security, and can be used to achieve goals of all types. For packet filtering, the network infrastructure ensures that some or all packets are forwarded through infrastructure members that perform filtering (they may also perform other functions such as forwarding). A *filter* looks for packets that satisfy its *filtering criteria*. On finding such packets, the filter takes some action as an exception to or as an addition to merely forwarding them.

In this section we look at how packet filtering works in a single network—or in a layer of homogenous bridged networks such as the Internet. In §2.4 we will return to the compositional view, considering how packet filtering interacts with other network mechanisms, and where it should be placed in a compositional network architecture.

The first subsection of this section gives an overview of basic packet filtering. Subsequent subsections explore major issues in more depth.

2.2.1 Filtering basics

The oldest filters are *firewalls*, dedicated so-called "appliances" positioned at or near the edges of a network. Their filtering criteria are predicates formed by combining mainly atomic predicates on the values of IP and IP-session-protocol header fields. Their function is to drop disallowed packets. For example, suppose that a firewall is intended to allow only outgoing Web accesses, which of course require outgoing DNS queries. The direction of a packet (inbound or outbound) can be determined from its source and destination addresses or from the link on which it arrives. The firewall might be configured with these four rules:

1. Drop all outbound TCP packets unless they have destination port 80.
2. Drop all inbound TCP packets unless they have source port 80 and the TCP ACK bit is set.
3. Drop all outbound UDP packets unless they have destination port 53.
4. Drop all inbound UDP packets unless they have source port 53.

In the second rule, the ACK bit indicates that this packet is an acknowledgment of a previous packet, meaning that it is not a TCP SYN packet.

These rules are sufficient for the purpose if all packets through the firewall obey the TCP protocol exactly, but of course an attacker may not be so polite. A safer approach would be to make the firewall stateful by having it maintain a table of all ongoing TCP connections. Then the second rule above would be replaced by "Drop all inbound TCP packets unless their source and destination addresses and ports identify them as belong to an ongoing TCP session." Stateful firewalls are often combined with NAT boxes, because NAT boxes sit at the edges of networks and already maintain tables of ongoing sessions.

This brief description contains or implies answers to the basic questions one should ask about a filtering mechanism, namely:

- What are the filtering criteria? For firewalls, configurable predicates on IP and IP-session-protocol header fields. If the firewall is stateful, the predicates can refer to a table of ongoing sessions.
- What actions are taken by the filters? Firewalls either drop or forward ("accept") packets. In addition to these actions, a filter can record packets, raise an alarm, or divert the packets for further analysis. If there is uncertainty about the packets, a filter can rate-limit them or downgrade their forwarding priority rather than dropping them. Rather than dropping session-initiation requests, a filter could reply to them with refusals, which would discourage retries. A refusal to a TCP SYN (request) is a TCP RST (reset). A refusal to an HTTP request is an error code.

- Which packets are filtered, which filter do they go to, and how are they steered into their filter? Firewalls are located at network edges, where every packet going into or out of the network necessarily passes through them. If a firewall is stateful, it is crucial that all packets of a session pass through the same firewall. This property is called "session affinity."

- How are the filters themselves, as potential traffic bottlenecks, protected from DoS attacks? Often firewalls are large machines, with capacity sufficient to handle all their network's traffic, even during a flooding attack. Because these firewalls are dedicated machines, without many programs or control interfaces, they cannot easily be penetrated by malware.

Stateless firewall functionality is sometimes implemented in forwarders rather than separate appliances, in which case the rules are called "access control lists."

For more sophisticated filtering (see §2.2.2), networks often use commercial products known as "intrusion detection systems" and "intrusion prevention systems." The difference is that detection systems only raise alarms, while prevention systems automatically take action against suspected attacks, such as dropping or rate-limiting packets. It might seem that automatic action is always better (it is certainly faster), but there are good reasons for keeping operators and network customers in the decision loop. If a suspected attack is a false positive, or even if its source is uncertain, much legitimate traffic may be dropped. If an operator deploys additional resources on behalf of an enterprise customer that is under attack, the customer will have to pay for them. The defense against a suspected attack may even be a counter-attack, which is wrong and even dangerous (in a military setting) if not well-justified.

2.2.2 Filtering criteria

Filtering criteria choose the packets on which a filter takes action. The problem of finding filtering criteria is somewhat different for the two major purposes of filtering.

If the purpose of filtering is to prevent or spy on specific communications, then the filtering criteria must describe the specific communications. However, many of these communications are not as specific as we would like. "Signature-based" filters such as spam filters, virus scanners, and parental filters look for keywords, sometimes keywords in specific positions, and other known attack patterns. They can also be stateful, and check whether protocols are being followed. These filters can be valuable commercial products because of the intellectual property in their filtering criteria. Like all security software, to be effective, they must be kept up-to-date. Even so, they cannot detect new attacks, and may be fooled by minor variations on old attacks. For one example, the attacker might fragment packets to hide their resemblance to the signature. For another example, a keyword in email text can be spelled creatively.

One advantage enjoyed by filters for preventing specific IP communications is that attackers cannot usually hide by spoofing. If the attack requires communication in both directions, then the attacker's source name must usually

be correct. (In more sophisticated attacks, the attacker can give a false source name and still receive return packets through route hijacking (§2.3.3) or packet sniffing.) Emails (in email application networks) are always one-way, however, so source names can be false.

If the purpose of IP filtering is to prevent flooding DoS attacks, it must contend with the fact that packet source names can be false. On the other hand, at least having a false source name is a straightforward packet-filtering criterion, if the filter can detect it. "Ingress filters" in IP networks check incoming packets to see if the prefixes of their source names match expectations. "Unicast reverse path forwarding (URPF)" in a forwarder accepts a packet's source name as valid only if its forwarding table specifies forwarding *to* the source name on the same two-way link on which the packet arrived. Unfortunately URPF cannot be used in the core (high-speed backbone) of a large network, because routes there are not necessarily symmetric. The Accountable Internet (see sidebar) is a network design with the principal goal of filtering out spoofed packets.

With the possible exception of their source names, the packets of a flooding DoS attack will look benign, so filtering criteria are computed by statistical algorithms. These algorithms look for anomalies, *i.e.,* variations from normal patterns of bandwidth use, protocol use, and other traffic attributes. Needless to say, there is a great danger of detecting too many anomalies ("false positives"), in which case many legitimate packets may be filtered out. Also needless to say, there are widespread hopes that machine learning will improve the precision of anomaly detection.

In general, the quality of filtering criteria is a limiting factor in the use of filtering to handle IP flooding attacks. The mechanisms for hiding attackers (§2.1.1) are effective. Almost all filtering works on the outermost IP header, regarding the rest of the packet as payload, so layering and encryption in the payload can conceal the true nature of the traffic. Because of these limitations, much of the research on DoS attacks aims to make filtering criteria precise by recognizing certain packets as desirable and rejecting all other packets. We'll call this approach "positive filtering" because the default action on a packet is to drop it, and matching a filtering criterion allows the packet to be delivered. In addition to precision, positive filtering has the advantage of preventing flooding DoS attacks, rather than reacting to them well after they have begun. Positive filtering is complex enough to deserve a section of its own.

2.2.3 Positive filtering

At least one kind of positive filtering is familiar and normal. Some email filters drop emails unless their source name is already a known correspondent of the destination.

A relatively simple kind of positive filtering is performed in a private IP network with software-defined control, as in Ethane [5]. An Ethane controller is a central network member with very complete knowledge of its network, especially the user members. For each user member the controller knows the IP and MAC addresses, the forwarder port to which the member is directly attached,

and the user of its machine. It also has policies governing which user members can reach which user members, the session protocols they can employ, and the middleboxes the sessions must pass through. When an Ethane forwarder receives the first packet of a session, it sends the packet to the controller, which uses its knowledge and policies to choose to either allow or disallow the session. If the session is allowed, the controller installs a tuple for it in the forwarding table of every forwarder in the path of the session.

Other proposals for positive filtering, including TVA [23], apply to mixed public-and-private IP networks. In this section we will concentrate on two proposals that together fit into an interesting and useful pattern.

Concerning Secure Overlay Services (SOS) [15] and Mayday [1], there are two important functional questions:

- What are the criteria for allowing or disallowing sessions?
- Where and how are good packets recognized and bad packets dropped?

With respect to criteria, SOS is intended for use during an emergency situation, when networks are so congested that even benign ordinary traffic must be dropped. The only allowed packets to a given destination come from a few pre-configured sources used by emergency responders. Mayday is a generalization of SOS providing for any approval criteria implemented by a certain class of network members (see below).

For recognizing good packets and filtering out all others, both SOS and Mayday rely on an overlay IP network of trusted, cooperating members—in a particular deployment instance, these members might not belong to the AA of a particular network, but instead might belong to an enterprise or peer group whose machines cooperate for mutual protection. In addition to the overlay network, a potential target must be surrounded in the Internet underlay by a ring of ordinary packet filters. These ordinary filters, such as firewalls or filtering forwarders, must have the capacity to handle flooding DoS attacks, and must be configurable by overlay machines or by people representing the overlay. The general idea is that good packets are transmitted to the target through the overlay, and the links of the overlay are implemented by the Internet underlay. The packet filters in the underlay can recognize which packets are also traveling through the overlay, and drop all other packets.

Figure 1 is a graph view of the physical arrangement. All nodes are members of Internet networks, either the protected target's network or other networks bridged to it. Note that all Internet paths to the target go through the filters. Note also that overlay machines can be located anywhere, close to the target or far away from it.

In designing an overlay network for positive filtering, there are three important choices to be made. SOS makes specific choices, while the Mayday paper points out that there are other choices, and evaluates some combinations of them. We now explain the three choices.

Source authentication. This choice concerns how an endpoint authenticates itself to the overlay as a source of legimate packets. In SOS the source is an

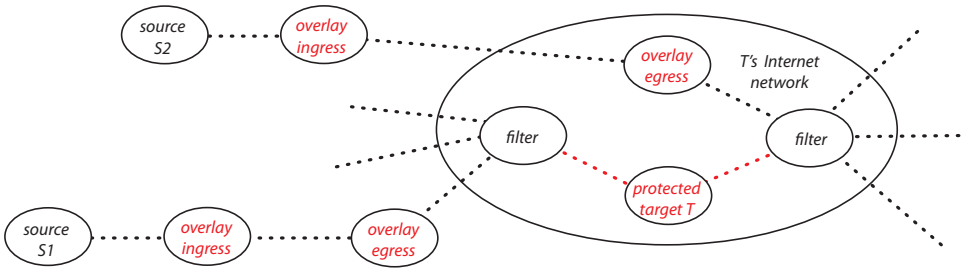

Figure 1: A graph view of Internet members involved in overlay-based positive filtering. The members named in red are on overlay machines, *i.e.,* their machines also have interfaces to the overlay network. The paths in red are the *only Internet paths* to the protected target.

overlay member, *i.e.,* it has special software. It creates a secure link to an overlay ingress member, using IPsec with endpoint authentication. SOS source members know the Internet names of many ingress members, well-distributed so that they cannot all be overwhelmed by flooding attacks.

Mayday emphasizes an option that is architecturally more complex, but has broader applicability because the source need not be an overlay member (both Figures 1 and 2 depict this option). In this option packets from a source to target name T are routed to some proxy that is an ingress member of the overlay. The proxy accepts the TCP session, and can then authenticate the source by asking for a user name and password associated with the target service. If the source is authentic, the proxy makes a TCP session *through the overlay* to the target; these two TCP sessions then become two parts of a compound application session.

Lightweight authenticator. Figure 2 is a session view of allowed access to protected target T. The last overlay hop between an egress member of the overlay and the target is implemented by an underlay path that goes through a filter. The lightweight authenticator is the attribute of underlay packets from egress member to target that causes the packet filter to recognize them as overlay packets and allow them to pass. The simplest lightweight authenticator is the IP name of the egress member (here E) in the source name of a packet; this is what SOS uses. Other authenticators proposed by Mayday include the destination port, destination name, and other header fields whose contents can be manipulated by the egress member.

The critical property of a lightweight authenticator is that it must be a secret—if attackers knew it, they could simply send underlay packets that match it. You might think that the destination name is the worst possible authenticator, but it can be a good one if the underlay name of the protected target is different from its overlay name, as shown in Figure 2, and if it can be changed easily and frequently by local control in the target's network.

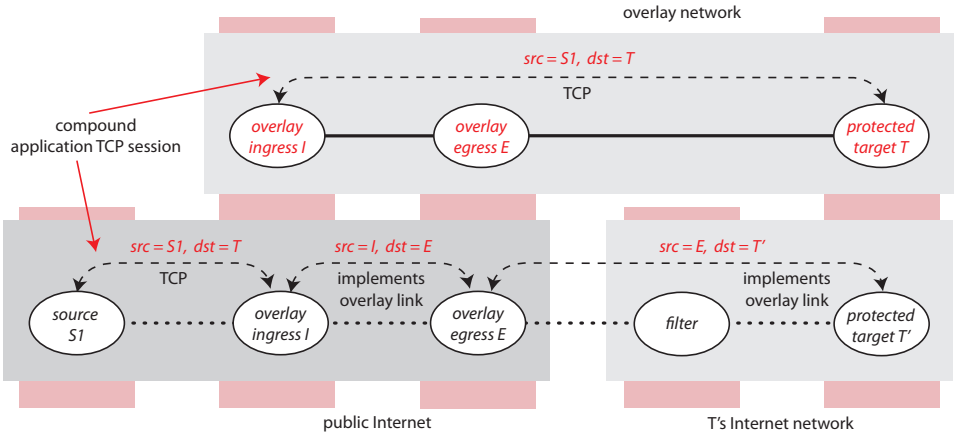

Figure 2: A session view of overlay-based positive filtering, illustrating the following options: source is not an overlay member, target has different names in overlay and underlay, routing is singly-indirect.

Overlay routing. The identities of authenticating ingress nodes are fairly public, as all packets to T must be routed to or destined for them. The purpose of having a full overlay network with its own routing (as opposed to having ingress nodes only) is to vary and hide the paths of packets between ingress members and the target. This keeps attackers from flooding the paths to the target rather than the target itself. It also keeps the identities of egress nodes secret, which is indispensable if the lightweight authenticator is the name of an egress node. SOS uses egress names as authenticators, and also employs complex routing through the overlay based on distributed hash tables. The paths are long and introduce considerable delay.

Mayday takes the position that effective overlay routing can be much simpler, with options including no routing at all (ingress and egress nodes are the same), and singly-indirect routing (one hop between ingress and egress nodes, as in Figure 2). The Mayday paper reports on analysis showing that certain combinations of overlay routing and lightweight authenticator provide "best cases" for trade-offs among performance and security. For example, designers who want moderate levels of both performance and security should use singly-indirect routing with any authenticator other than source name.

2.2.4 Filtering resources

When filtering is used as a defense against flooding attacks, the filters and the network paths to them must have sufficient resources to handle all of the attack traffic. Thus resources and scalability are important aspects of the design of packet-filtering mechanisms.

In a layer of homogenous bridged networks such as the Internet, we can visualize the graph of all network paths leading from sources to an attack target as a giant tree. Filters should be arranged so that there is a filter (or sequence of them, if filtering is pipelined) on each path. Considering this tree, there are general trade-offs between placing filters closer to the target or farther from the target. The advantages of placing filters closer to the target are:

- The root of the tree is at the target. Obviously, there are fewer paths to cover close to the target, and fewer packets overall for the filters to process.
- Usually the responsibility for protecting the target lies with the AA of the target's network, so the AA has an incentive for deploying resources in its network, close to the target.

The advantages of placing filters farther from the target are:

- If filtering is farther from the target, the damage done by attack traffic is lessened, because attack traffic is carried for shorter distances along fewer links. Note that the damage of a flooding attack is not limited to the intended target, because traffic to many other destinations will also suffer because of congested links.
- If filtering is farther from the target, it is closer to the sources of attack traffic, and may have more information about it. For example, an ingress filter in an IP access network knows the prefix of all genuine source names, so it can filter out packets with false source names. The access network sees all of a suspected source's traffic, so attack patterns are more likely to be detectable. An access network may also know more about the type and reputation of its sources (device type is relevant because some mobile operating systems and vendor hardware are more easily penetrated than others). More precise filtering means less collateral damage.
- When there is a major flooding attack, it is usually focused on a small number of targets. Very often, the attack packets are coming from a botnet, with a large number of sources well-distributed across the public Internet. So the total amount of available filtering resources near sources greatly exceeds the total amount of resources available near targets.

A third option, filtering in the topological core of the network, is never used because the core is a region of high-speed links and high-speed routers handling large flows of packets. Routers would have to filter at line rate, based on filtering rules stored in expensive Ternary Content-Addressable Memory, and the hardware's capacity to store rules would not be adequate for the number of rules required.

Unfortunately, the many advantages of filtering near sources are balanced by two major disadvantages:

- Networks may not have sufficient incentive to use their resources to protect targets that are remote from them. (However, this is making a big assumption, that topologically distant networks are also distant administratively. If the administrative distance is small, then this disadvantage disappears.)

- Even if source networks are willing to cooperate with target networks, the necessary coordination is not easy. Infrastructure machines such as routers, controllers, and policy servers must cooperate on attack diagnosis as well as filtering. (Before the recent advent of programmable routers, this would not even have been possible.) Control communication between networks must be secure, because attackers could abuse it. Yet ordinary endpoint security may not be applicable, because the networks do not know the identities of forwarders in other networks, nor do they know which forwarder is on which path. Finally, even when communicating correctly with another network, the target network cannot necessarily trust it.

There are proposed solutions to all the problems of incentives and control [3, 7, 19, 22], but they are not simple. Historically, cooperation between networks with different AAs has been scarce [10].

The proposals for moving filtering toward the sources of attack traffic date from the early 2000s. In the 2010s cloud computing has advanced so far that most filtering is performed in clouds on behalf of target networks. In clouds the filters are virtual machines, and the number of filters expands and contracts with fluctuations in load. It is also practical to employ sequenced filters, directing packets that early filters find suspicious to later filters with more detailed screening.

2.2.5 Proxies

Proxies, and compound sessions formed by these proxies, are commonly used to evade packet filtering. This works because the header fields of the packets in the component simple sessions of a compound session can be completely different. Although joinboxes can also form compound sessions, they are not endpoints of session protocols, and therefore cannot do all the manipulations we will see proxies doing in the remainder of this session.

Perhaps the oldest example of such a proxy is an "application gateway," which is installed in a private IP network for the benign purpose of evading the too-simple filtering imposed by the firewall. For example, an enterprise firewall may block all outgoing sessions except Web accesses. However, the enterprise may also wish to allow outgoing sessions of another kind, when they are initiated by specific users. The firewall cannot enforce this policy because it does not know the mapping between internal IP names and users (and the mapping may not even be static).

An application gateway for the application, for instance Telnet, solves this problem. To use it, a user initiates a Telnet session to the application gateway

inside the enterprise network. By means of an extension to the Telnet protocol, the user supplies a password to authenticate himself to the proxy, and also the name of the real Telnet destination. The proxy initiates a Telnet session to the real destination outside the enterprise network, and joins the two simple sessions in a compound session. The enterprise firewall allows outgoing Telnet sessions from the application gateway only.

Proxies are the principal tool for providing users with privacy, anonymity, and the ability to evade censorship. We will show how this works in three stages.

The first stage is simplest. If, for example, a user wishes to access a Web server with some privacy and anonymity, the user's browser requests from the IP interface on the user machine a "proxied TLS" session with a friendly proxy outside his home network (Figure 3). A "proxied TLS" session is just like a normal TLS session except that: (i) instead of looking up the domain name *dangerous.com* and using its IP address as the destination of the session, the IP interface uses the proxy's address as the destination of the session; (ii) it expects and verifies the certificate of the proxy, not the Web site.

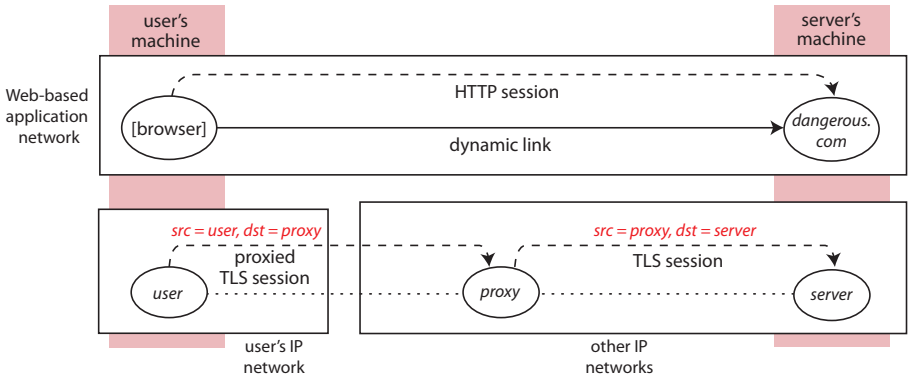

Figure 3: A proxied TLS session protects the user's privacy in his home network, and anonymity at the Web server.

After the proxied TLS session is set up, it appears perfectly normal to the user's machine. The proxy, however, decrypts the HTTP request, and uses it to make another TLS session with the actual server. After this the proxy relays packets between the two components of the compound TLS session. Because of the compound session, the IP network of the user does not know what the user is connected with, and the final destination does not know who the user is.

One remaining disadvantage of the first stage is that the user has no privacy from the proxy. The other disadvantages come from the fact that the addresses of helpful proxies must be publicly available (so users can find them) which means that they are available to the user's adversaries as well. Most importantly,

if the user's IP network is censoring the network activity of its users, it can simply block packets addressed to external public proxies.

The second stage of proxy use is intended to evade censorship in a network, such as a national ISP, that seeks to limit or spy on the network activity of its users. There are several similar proposals [11, 12, 21], all using proxies, but in a way that still works even if the censoring network is blocking access to known proxies. As a representative of these proposals, we will present Cirripede [11].

As shown in Figure 4, the ISP of the Cirripede client is filtering out packets from the client to certain Web servers, represented here by the "covert destination." The client cannot evade this censorship by using a false source name, because then replies from the Web server will not be delivered to the client (also, the network may be blocking *everyone's* access to the site).

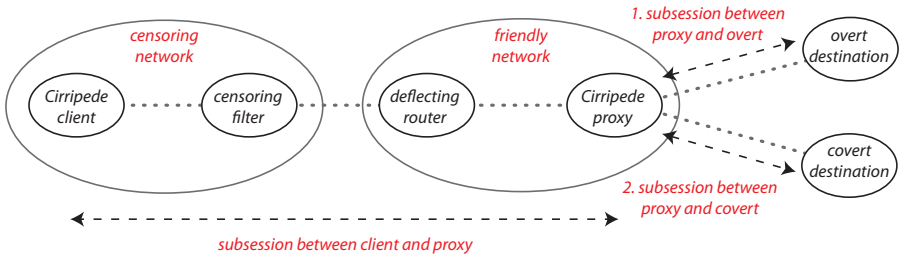

Figure 4: Subsessions of a session between a Cirripede client and a covert destination. In the subsession on the left, names in the IP header are those of the client and overt destination; packets from the client are deflected to the proxy as an exception to normal forwarding.

For Cirripede to work, there must be a network operated by a friendly AA on the path between the client's ISP and many destinations, including the covert destination. The client must first signal to the friendly network that it wishes to use Cirripede for a time interval. To register as a Cirripede client, it must use a "covert channel" of communication, one that its ISP is unlikely to recognize. Covert channels are constructed from attributes of a packet stream that the sender can control and observers of the packet stream are unlikely to notice, including packet timing, order of TCP packets, unused packet fields, and pseudo-random packet fields. (Covert channels are always low-bandwidth, and vulnerable to detection if popular, which is why they must be used sparingly.) To signal covertly that it wishes to register with the friendly network, the Cirripede client puts a secret number in the pseudo-random initial-sequence-number field of a TCP SYN packet. The packet is intercepted by a router in the friendly network on the path from the client, and converted to a registration for the client. As a result of the registration, a special "deflecting table" in friendly routers over-rides normal routing. When subsequent packets from the client reach these routers, they are sent to a Cirripede proxy.

The Cirripede proxy will help the client access the covert destination, but

it takes a complex session protocol to do this safely. First, the client requests a TLS session with an overt destination—one that the client is allowed to reach—and completes a TLS handshake with it. During this phase the proxy is present in the path but takes no active part. Even if the censoring filter is monitoring every packet of the TLS handshake, it will see nothing out of the ordinary. After the TLS handshake all packets to and from the client will be encrypted, so their content need not be constrained. During the entire TLS session, the packets seen by the filter will have the overt destination in their source or destination field.

Next, the proxy will end the session between itself and the overt destination, and inform the client that the proxy is in the path of the session. The client will respond by sending it the name of the covert destination it actually wishes to reach. The proxy initiates a TLS session to the covert destination, and afterwards relays packets between the client and covert destination.

Just as the censoring authorities could learn about friendly proxies and block their IP addresses, could the authorities learn about the helpfulness of the friendly network and refuse to bridge with it? The assumption behind Cirripede is that the friendly network is on the censoring network's path to large parts of the Internet. The prediction is that the censoring network will not cut it off, because the effects would be too public and too draconian.

The remaining vulnerabilities are that the user still has no privacy from the proxy, and that that Web site or other destination knows that the user is connecting to it through a proxy (again, because proxies are publicly known). The latter issue will be discussed in §2.4.3. Privacy from the proxies can be obtained by using the Tor service.

Tor [8] is a well-known public service for providing users with private, anonymous sessions to Web sites and other services. Volunteers worldwide lend their machines' resources to act as Tor proxies. Tor proxies form an overlay network with its own randomized routing (as in §2.2.3) to carry the users' traffic. Most importantly, the initiator of a Tor session chooses proxies for the session, and each Tor proxy has a public key. Packets of the session are encrypted as many times as there are proxies, working from the key of the last proxy to the key of the first. Each proxy decrypts one layer of encryption, so that it knows the names of adjacent proxies in the chain, but nothing else.

2.3 Security that is not packet filtering

The majority of infrastructure security is implemented with packet filtering, which is a general mechanism that can be used for many purposes. In this section we review a few techniques for infrastructure security that are *not* packet filtering.

2.3.1 Resource replication

Current network technology, has made it both feasible and popular to defend services against resource attacks by replication of resources, so that there are

enough resources to withstand attacks. This is easiest in a cloud, where a service under attack can quickly be granted more virtual machines.

It is even better if resource replicas are geographically distributed, so that some replicas can be reached when other parts of the network are too congested. Because attacks on DNS servers are so common and damaging, it is especially important to have distributed authoritative DNS servers for popular domain names. Queries are distributed across the replicas by means of DNS anycast or IP anycast. If there are five replicas sharing the load and one has been overwhelmed by an attack, DNS or IP anycast may not be dynamic enough to redirect queries away from the failed replica, but at least queries directed by anycast to the other four will succeed.

Resource replication works best when replicated data is fairly static, and it is not feasible when queries update the data. (Note that this remark is specific to the context of replicating data as a defense against flooding attacks in the public Internet. In more private distributed systems, data is frequently replicated for reliability, and there are many protocols for making data that has been updated at a single replica consistent across all replicas.) In cases where data is dynamic or updated by queries, it can be distributed across multiple sites by sharding, *e.g.,* by partitioning the keys of a key-value store across sites. This will increase the total resources available, and also the expected availability of an arbitrary key.

2.3.2 Reduction of amplification

Another defense against resource attacks is to reduce the amplification factor on which they depend. For example, It has become common for servers to defend themselves against SYN floods (§2.1.1) with "SYN cookies." In this defense, the server returns a SYN+ACK packet with a specially-coded initial sequence number (the cookie). It then discards the SYN, using no additional resources for it. If the SYN was an attack, it has not been amplified. If the SYN was legitimate, on the other hand, it will elicit an ACK from the initiator with the same initial sequence number incremented by one. By decoding the sequence number, the server can reconstruct the original SYN and then set up a real TCP connection.

A flood of DNS queries is amplified when servers query other servers. A very effective defense against these attacks is longer times-to-live for cache entries, perhaps 30 minutes, in recursive and local DNS servers [20]. If local entries are cached longer, there will be fewer queries and retries made to authoritative servers. There are many good reasons for DNS cache entries with short times-to-live, but these can be changed as an adaptive measure during attacks. The same idea would work for many other services with caching.

2.3.3 Endpoint security for control protocols

Control protocols are used to maintain and distribute network state. It should be no surprise that unauthorized update or distribution of network state can be

used for malicious purposes. The most well-known attack of this kind is "BGP hijacking," in which an attacker advertises an IP prefix it does not own as a way of drawing packets for that prefix toward itself.

The defense for a control protocol is endpoint security. For instance, Border Gateway Protocol Security is a security extension to BGP that provides cryptographic verification of messages advertising routes. Similarly, Domain Name System Security Extensions protect DNS lookups by returning records with digital signatures.

The use of TLS or ESP is not always possible for control protocols. An endpoint may not have a certificate or other credential to prove its identity. The protocol might require high-speed, high-volume operation. Or, the protocol might simply be too old to incorporate endpoint security, even if it is feasible.

In these cases there are lighter-weight measures that can help. Network members that make requests should keep track of their pending requests and not accept unsolicited replies. Replies should be checked for credibility, whenever that is possible. Most effectively, a network member can include a nonce or random field value in a session-setup message or request. Subsequent messages of the protocol must have the same nonce or random value, so that no attacker without access to the previous messages of the session can send messages purporting to be part of the session. Without the nonce, an attacker could do something to trigger a query, then send a spurious answer to the query.

2.3.4 Very special-purpose techniques

In special cases, undesirable communication can be prevented by making a user member unreachable, by altering either the network's directory or the network's forwarding. For example, an illegal gambling site can be made unreachable in the public Internet—at least to those who do not know its IP address—by removing or falsifying its DNS entry. This is a special case because, although the gambling site is connected to the public Internet, officials want to prevent *all* communication with it.

2.4 Compositional infrastructure security

This section is concerned with the interactions among infrastructure security in single networks, composition operators, and other aspects of networking.

2.4.1 Bridging

When networks are bridged, they have either *shared links* or *shared members*. If a member is shared it belongs to both networks, and may play a different role in each. For instance, a gateway to a private network may be shared between the private network and its ISP's network; in the private network it is an infrastructure member, while in the ISP's network it is a user.

For infrastructure security, the most prominent threats occur among the many bridged networks of the public Internet. There is much less concern about

the security of Ethernet LANs, particularly wired Ethernets, because they are usually isolated—their member machines usually connect to the outside world through IP networks at a higher level. Wired Ethernets can be attacked, but the attackers must have some kind of physical access to the machines connected to them [17]. In addition to attacks by insiders, a visitor to an enterprise might plug into an unused wall socket or switch a cable from one machine to another. A determined attacker might even even drop a memory stick with a company's logo in the company's parking lot; someone who finds it might plug it into a company computer to see who it belongs to.

2.4.2 Layering

A machine could be a member of several independent networks and, because of fate sharing, be attacked through any of them. Most often, however, each packet arriving at a machine is being transmitted through multiple layered networks simultaneously, for example an Ethernet LAN, an IP network, and an application network. Amplification of a flooding attack, or damaging processing of a malware packet, can take place at any of these levels. Filtering can also take place at any of these levels.

First we consider the networks layered below the filtering network and implementing its virtual links. If the filtering is to achieve what we assume it is achieving, then for all links on paths between a filter and a protected target, it must be true that no packet is received on the link that was not sent on the link. In other words, lower-level networks do not inject packets into the implementation of the link.

This might seem like a fanciful concern, but it is very real in clouds. The filtering network might be a tenant of a multi-tenant cloud, in which case its links are virtual channels implemented by sessions in a lower-level cloud network that is shared among tenants. If the cloud network does not isolate tenants properly, then cloud-network members of another tenant might pretend to belong to the filtering network. They might insert packets into filtering-network sessions in the cloud network, or manipulate packets in them before they are delivered to the session endpoint. Even if the other tenant is an honest enterprise, its virtual machines may have been penetrated by attackers (which is easy to do, see the sidebar on botnets). Even though the AA of the filtering network is associated with the tenant, the integrity of its links is the responsibility of the cloud network, whose AA is the cloud provider. In another scenario, a penetrated Ethernet can also inject packets into the links of networks layered on it [17].

It is common to deploy IP-based intrusion detection systems that look into the payloads of IP packets for attack signatures at higher levels, for example signs of malware at particular locations in the payloads. These systems have the limitation that they are assuming a known and fixed layer architecture between themselves and the endpoints they are protecting.

In many ways it makes more sense to include filtering middleboxes in session paths in each network separately, thus making no assumptions about which

other networks a packet is traveling through. A network-specific filter, particularly for an application network, can rely on far more knowledge of the application and network members. Today virtualization technology makes this approach far more practical than it was in the past. Of course, higher-level filtering can augment lower-level filtering, but cannot replace it.

The overlay architectures in §2.2.3 offer another attractive possibility. If an endpoint machine should receive *only* packets transmitted through a higher-level (overlay) network, then the overlay can filter, and the implementation underlay network can be configured to allow only those underlay packets that are also overlay packets.

2.4.3 Middleboxes

An important interaction between endpoint security and infrastructure security has already been covered. Packet filters are middleboxes, so all the observations about encryption and middleboxes (§1) are relevant.

In general, a middlebox can alter a packet stream that passes through it. For this reason, only trusted middleboxes should be placed in packet paths between a packet filter and a target it is intended to protect.

The interaction between filtering and proxies is of paramount importance. With proxies, a session can consist of a chain of multiple simple sessions, each of which has completely different headers. As we have seen, if the purpose of filtering is to prevent flooding attacks, then there is some flexibility as to where the filtering can be located. So effective filtering must be located where an attacker's sessions can be identified by the headers of the simple sessions in that locality.

If the purpose of filtering is to block or spy on specific communications, then it can be evaded by making the simple session of a compound session look innocuous in the blocking AA's network. This statement covers not just evasion of censorship in a user's access network, but also supposed spying by the proxies themselves (one of the threats addressed by Tor). The AA of a user's access network can exercise some control over this by choosing not to bridge with networks harboring proxies, but this mechanism of control is limited by the AA's need to maintain connectivity with all or most of the global Internet, with reasonable performance and cost.

Unfortunately the Tor mechanism for privacy has one deficiency with no current solution. At the far end of the session, the acceptor of the session can know that Tor is being used, because there are readily accessible and regularly updated lists of Tor nodes (necessary so that prospective users can find them). Fraudsters, spammers, and other criminals are big users of Tor, along with law-abiding people in need of privacy. Consequently an increasing number of services are rejecting or otherwise discriminating against Tor users [16].

2.4.4 Routing

Wide-area routing frequently creates different paths for packets traveling in different directions between the same two endpoints. Even packets traveling in the same direction may be spread across multiple paths because there has been a failure in one of the paths, or a need for better load-balancing.

Thus routing requirements can conflict with the need for session affinity (§2.2.1), *i.e.,* with the need to have all packets of a session pass through the same filter. These needs are easiest to reconcile within a cloud, where paths come together and there can be a mechanism for sending all packets of a session to the same virtual instance of a filter.

2.4.5 Session protocols

SYN cookies (§2.3.2) are very clever, but like most clever solutions, they come with hidden limitations. A server using SYN cookies effectively drops all optional information in TCP SYN packets, which means that any network capability or feature relying on extensions to TCP is disabled. Note that many servers using SYN cookies are Web servers, and many new features relying on extensions to TCP (such as Multipath TCP, just to name one example) have improved Web access as a major use case. These issues are also discussed in the chapter on *Session Protocols*.

There is another defense against SYN floods that is less efficient than SYN cookies, but comes with fewer limitations. This defense uses a proxy in the path to a Web server that stores and responds to SYN packets, but does nothing else with a SYN packet until it receives the ACK that completes the handshake. On receiving the ACK, the proxy forms a new session by sending the SYN to the server, and subsequently acts as a transparent proxy between the two sessions. If the proxy does not receive a timely ACK, then the SYN packet was part of an attack or the client has failed, so the proxy drops it.

3 The verification challenge

There is growing interest in verifying the correctness of networks, most importantly with respect to security. For example, there has been significant progress on verifying the forwarding tables of an IP network—or the algorithm that computes them—to ensure that they satisfy reachability and access-control policies [2, 4, 13, 14, 18].

It should be abundantly clear from this chapter, particularly from the sections covering the interactions among composition operators, security mechanisms, and diverse network mechanisms for requirements other than security, that this work is necessary but not sufficient.

If this is not convincing enough, one need only look at the hundreds of new papers on security published every year. Each responds to a specific known or suspected security threat with a specific defense—which is often complex. It is hard to believe that network operators will ever be able to deploy all of these

defenses, and troubling to think that they will have to choose some arbitrary subset of them. Furthermore, many of these security papers are excellent. They use careful and subtle reasoning to enumerate possible attacks and discuss which ones their defenses should hold against. But there are so many that one is driven to say, "What an insightful list! But how do I know that they thought of *everything*?"

We feel that there is an urgent need to start integrating and unifying this knowledge, and that compositional architecture provides a framework for doing so. We are beginning by analyzing results from a large number of research sources, and unifying them through use of the model. Further steps will leverage the structure and modularity of the model to reduce the "attack surface" of networks that satisfy today's requirements, so that they can have property guarantees built into their software and hardware.

Exercises

3. This exercise concerns overlay-based positive filtering of TCP traffic. The lightweight authenticator is the combination of destination TCP port and destination name, where destinations are found in a public /24 subnetwork of the Internet. There are several ways that an attacker can probe a subnetwork, finding out whether or not a SYN packet reaches a server [1]. Let's say that a probe requires 7 packets on the average, probe packets are 130 bytes on the average, and the subnetwork is served by a 100 Mbps Ethernet. What is the minimum time it would take for an attacker to check all possible lightweight authenticators, to find the one that gets through the filters?

4. Explain in detail how VPN technology fits the overlay pattern for positive filtering. Are there differences from SOS and Mayday as examples of the pattern?

5. Read the SOS paper [15] and assign a unique name to each object or object type referred to in the SOS architecture. For each of these object or object types, what is the set of names used for it in the paper? What do you think of the technical writing in this paper?

Sidebars

Sidebar: Botnets

A botnet is not a network according to the strict definition in this book, but rather a group of user members of the public Internet. A botnet can be assembled by an illegal enterprise, and rented out to attackers for small fees.

A machine becomes part of a botnet when it is penetrated and infected with a particular kind of malware. This is all too easy for certain large classes of machines, which is why botnets of millions are common.

Machines are most easily penetrated when their manufacturers install default passwords in their software, and users connect them to the public Internet without changing their passwords. Attackers simply try IP addresses until they find such machines. Classes of such machines and their default credentials are posted on the Web for attackers to use.

Internet of Things machines are particularly vulnerable, because they include devices such as home routers, digital video recorders, and baby monitors often operated by users who are not computer-savvy. Another vulnerable class is virtual machines in public clouds, because they are initialized with operating-system defaults and can be penetrated before their customer has access to them!

Once a machine has become part of a botnet, it behaves normally until the master of the botnet sends it a triggering signal. Once triggered, it begins to generate IP packets to contribute to a flooding attack.

References

[1] D. G. Andersen. Mayday: Distributed filtering for Internet services. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*, 2003.

[2] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker. Netkat: Semantic foundations for networks. In *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, January 2014.

[3] K. Argyraki and D. R. Cheriton. Active Internet traffic filtering: Real-time response to denial-of-service attacks. In *Proceedings of the USENIX Annual Technical Conference*, 2005.

[4] R. Beckett, R. Mahajan, T. Millstein, J. Padhye, and D. Walker. Don't mind the gap: Bridging network-wide objectives and device-level configurations. In *Proceedings of ACM SIGCOMM*, 2016.

[5] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. In *Proceedings of SIGCOMM*. ACM, August 2007.

[6] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: Defining tomorrow's Internet. *IEEE/ACM Transactions on Networking*, 13(3):462–475, June 2005.

[7] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP traceback. *ACM Transactions on Information and System Security*, 5(2):119–137, May 2002.

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

[9] Dyn analysis summary of Friday October 21 attack. https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/. Accessed 10 November 2018.

[10] M. Handley. Why the Internet only just works. *BT Technology Journal*, 24(3):119–129, July 2006.

[11] A. Houmansadr, G. T. K. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2011.

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

[13] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte. Real time network policy checking using Header Space Analysis. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, 2013.

[14] P. Kazemian, G. Varghese, and N. McKeown. Header space analysis: Static checking for networks. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, 2012.

[15] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of SIGCOMM*. ACM, August 2002.

[16] S. Khattak, D. Fifield, S. Afroz, M. Javed, S. Sundaresan, V. Paxson, S. J. Murdoch, and D. McCoy. Do you see what I see? Differential treatment of anonymous users. In *Proceedings of the Network and Distributed Security Symposium*. Internet Society, 2016.

[17] T. Kiravuo, M. Sarela, and J. Manner. A survey of Ethernet LAN security. *IEEE Communications Surveys & Tutorials*, 15(3):1477–1491, 2013.

[18] N. P. Lopes, N. Bjorner, P. Godefroid, K. Jayaraman, and G. Varghese. Checking beliefs in dynamic networks. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, 2015.

[19] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Schenker. Controlling high bandwidth aggregates in the network. *Computer Communication Review*, 32(3):62–73, July 2002.

[20] G. C. M. Moura, J. Heidemann, M. Muller, R. de O. Schmidt, and M. Davids. When the dike breaks: Dissecting DNS defenses during DDoS. In *Proceedings of the ACM Internet Measurement Conference*, 2018.

[21] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the network infrastructure. In *Proceedings of the 20th USENIX Security Symposium*, 2011.

[22] A. Yaar, A. Perrig, and D. Song. SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2004.

[23] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *Proceedings of SIGCOMM*. ACM, August 2005.