

Security and Robustness in the Internet Infrastructure

Krishna Kant¹ and Casey Deccio²

¹ George Mason University, Fairfax, VA, kkant@gmu.edu

² Sandia National Laboratories*, Livermore, CA, ctdecci@sandia.gov

Abstract. Security and robustness of the Internet infrastructure is essential for ensuring that the online services operate as expected. The essential components of Internet infrastructure include the Domain Name System (DNS) that provides translation between user friendly names and network addresses, inter and intra-domain routing at IP level, layer-2 switching, quality of service provisioning, and infrastructure security protocols such as IPsec, SSL, etc. In this chapter we discuss the security and robustness issues connected with these aspects of the Internet. One major cause of security and robustness challenges is the configuration complexity and the resulting misconfigurations that create security holes and undesired behavior. This chapter discusses the issues of network configuration robustness and points out challenges in several areas. This chapter does not address misconfigurations related to packet handling rules and policies as those are covered elsewhere in this handbook.

Keywords: Domain Name System (DNS), DNSSEC, Configuration Management, IPsec, Transport Layer Security, Public Key Cryptography, Routing, Border Gateway Protocol, Switching, Virtual LAN, Spanning Tree Protocol, Data Center, Fat-tree.

1 Introduction

The integrity of networking infrastructure is crucial in ensuring that the core network services involved in providing communication functionality work reliably, effectively, and without side effects. Some of the core networking services include resource addressing, packet switching/routing, packet classification/filtering, name resolution, end-to-end transport, and network Quality of Service (QoS) management. These issues are important not only in the wide-area network context but also for access networks on client side and the data center network on the server side.

* Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

A major source of threats to network services arises from the fact that every protocol and service needs to depend upon a set of configuration parameters that may be either inadvertently misconfigured or deliberately perturbed by a hacker. For example, successful application level communication will typically require correct configuration of DNS, TCP, routing tables, firewalls, NIC cards along the way, etc. As the underlying services and protocols become more numerous and complex, the issues of proper configuration become more serious. For example, secure communication would require proper settings for security protocols such as IPsec or SSL, and the increased complexity of DNSSEC makes it much harder to get it right than plain old DNS. The issues of proper configuration management permeate all aspects of computing, but this chapter only focuses on a few issues related to networking. Reference [30] provides a more in-depth consideration of security of configuration management in data centers, and discusses techniques for both the detection of misconfigurations and for hardening the configuration against attacks. Although, we shall summarize some of this work here, the focus of this chapter is on specific vulnerabilities associated with name resolution, routing, and network QoS management.

Maintaining the consistency and integrity of communications in the face of misconfigurations and erroneous/malicious updates is a challenging problem and involves many facets, but some significant ones are specifically left out here since they are addressed by other chapters in this handbook. In particular, we do not address misconfigurations related to packet treatment policies at firewalls and other packet filtering devices as those are covered in chapter ???. Also, our coverage is generally technology agnostic (e.g., independent of whether the links are wired, wireless, or optical). The security of wireless, sensor and ad hoc networks is already addressed in earlier themes of the handbook and thus not included here.

The outline of this chapter is as follows. Section 2 discusses the basics of the Domain Name System (DNS) and its numerous vulnerabilities. Section 3.2 then discusses various proposals to enhance DNS robustness, including DNSSEC, and the new set of challenges that DNSSEC brings. Section 4 discusses some popular end-to-end security protocols such as IPsec and TLS and points out issues in their deployment. Section 5 then focuses on the integrity of Internet routing including routing misconfigurations and border gateway protocol (BGP) robustness. It also discusses how to enhance robustness of BGP under large scale failure scenarios. Section 6 discusses security and integrity issues with Ethernet layer-2 which is of considerable importance in the context of data centers and metro-Ethernet. Section 7 deals with the very important emerging topic of configuration management security. Finally, section 8 concludes the discussion and points out a number of areas of future challenges.

2 Vulnerabilities in Domain Name Resolution

The *Domain Name System (DNS)* [43,44] is a globally distributed database containing mappings of *domain names* to Internet resources, particularly IP

addresses. Name resolution using the DNS allows users and applications to identify resources by name rather than address for usability and flexibility. For this reason it is essential that the DNS exhibits both availability and integrity. In this section, we discuss the basic operation of DNS and point out its numerous vulnerabilities.

2.1 Domain Name System

The DNS namespace is organized hierarchically, and each *domain name* is comprised of dot-separated labels that reflect its ancestry. For example, the ancestry of *www.foo.net* is: *www.foo.net*, *foo.net*, *net*. All domain names are descendants, or subdomains, of the *root* domain.

A *zone* is an autonomously managed piece of DNS namespace, typically administered by a single organization. A set of name servers are configured and advertised as *authoritative* for each zone. Each zone contains resource records (RRs), each of which has an owner name (e.g., *www.foo.net*), a time-to-live value (TTL, e.g., 3600), a type (e.g., **A**), and record data specific to its type. For example, the record data for an RR of type **A** (address) is an Internet Protocol version 4 (IPv4) address (e.g., 192.0.2.3). Some RR types, such as the **CNAME** (canonical name) RR, use another domain name as record data—a target for further lookup (e.g., RRs of the same name and type comprise a *resource record set* (*RRset*)).

Servers are designated as authoritative for a zone by including their domain names as the targets of RRs comprising the **NS** (name server) RRset corresponding to the zone. To properly handle delegation, these records must exist in the child zone, and also in its parent, as so-called delegation records. These RRs form the link between the parent and child, and proper function requires careful coordination between administrators of both zones.

To resolve a domain name a *resolver* iteratively queries *authoritative servers*. The resolver begins by issuing a query to a server authoritative for the DNS *root*, which refers the resolver to the servers authoritative for the top-level domain (TLD) of the domain name being resolved (e.g., *net*). The resolver follows the referrals downward in the namespace hierarchy until it receives a response from an authoritative source containing either an answer or a response indicating that the requested resource does not exist. In Figure 1 the process for resolving the domain name *foo.net* is illustrated.

DNS data consists of *resource records* (RRs), grouped together by their names and types as *RRsets*. A resolver may store an RRset in cache for the duration of the time-to-live value (TTL) associated with the RRset. A *zone* is an autonomously managed piece of DNS namespace, typically administered by a single organization. For each zone, authoritative servers are designated through use of the **NS** (name server) RRset for the zone. These servers answer authoritatively for names in their namespace or provide referrals for names in delegated namespace.

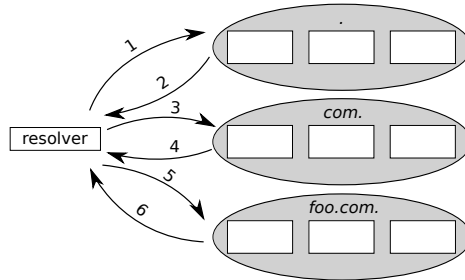


Fig. 1. The name resolution process for *foo.net*.

2.2 Dependencies in the DNS

Name dependencies in the DNS exist both as part of its hierarchical structure and as a result of explicit configuration. While these dependencies allow administrative flexibility to the DNS, they also add complexity, which can affect the availability and reliability of a domain name. Three specific DNS components introducing domain name dependencies are the following:

Parent zones A resolver must learn the authoritative servers for a zone from referrals from the zone's hierarchical parent. For example, *foo.net* depends on the *com* authoritative servers to provide the proper delegation records for the name in question.

NS targets The NS RR type uses names, rather than addresses, to specify servers authoritative for a zone, so a resolver must have the corresponding addresses to query the servers. In the case of NS target names that are subdomains of the referring (parent) zone (*in-bailiwick*), *glue* records may be introduced into the parent zone; for NS target names that are subdomains of the delegated zone, glue *must* be introduced into the parent zone. In either case, when glue exists, the addresses are made available by the parent in the referral, so there is no dependency. However, a resolver must independently resolve any other names. For example, if *ns1.bar.com* is among the NS target names for *foo.net*, a resolver obtaining such a referral must resolve *ns1.bar.com* to obtain its address.

Aliases If a name is an alias (i.e., corresponds to a CNAME RR), then to obtain the DNS for the name, a resolver must subsequently resolve the alias target.

DNS dependencies are transitive and may be modeled as a directed graph reflecting dependency relationships [15, 16]. Figure 2 illustrates name dependencies for the *foo.net* name. The edges between elliptical nodes represent the name dependencies of different types.

Edges between elliptical nodes (domain names) and rectangular nodes (IP addresses representing authoritative name servers) in Figure 2 represent *server dependencies*. Server dependencies stem from one of two different circumstances:

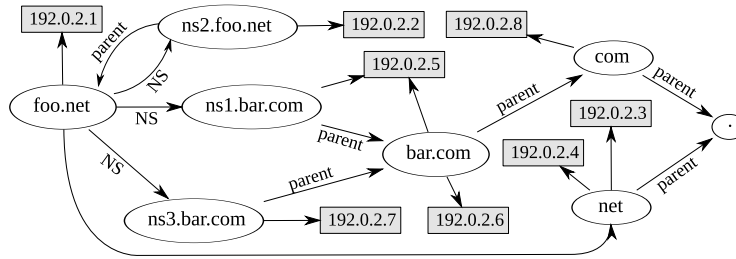


Fig. 2. The server dependency graph for *foo.net*. The gray, rectangular nodes represent name servers, and the oval nodes represent domain names. Edges between one node to another represent a dependency of a domain name on another name or a server.

a zone that has an in-bailiwick NS target name with a glue record (e.g., *foo.net* → 192.0.2.1); or a domain name that resolves to an Internet address (e.g., *ns1.bar.com* → 192.0.2.5).

Dependence for name resolution is modeled as a recurrence relation using the dependency graph. Each node representing a zone is dependent on both its parent zone and any one of its server or name dependencies. That is, its parent must be resolvable, so the delegation can be followed, and it must be able to query and receive a response from an authoritative server—either one provided by an in-bailiwick glue record or one learned by resolving the name of NS target independently. Each node representing the name of a server is also dependent on its parent zone, as well as the address to which it resolves.

This recurrence produces a logical tree for determining the combinations of server availability that are necessary for a particular domain name to be resolved, such as that illustrated in Figure 3 for the dependencies from Figure 2. The *foo.net* zone depends on the *net* zone (parent) and either 192.0.2.1 (address supplied with in-bailiwick glue record), *ns2.foo.net*, *ns1.bar.com*, or *ns3.bar.com* (other NS target names). The *ns1.bar.com* domain name is dependent on the *bar.com* zone (parent) and 192.0.2.5 (the address to which it resolves), and so on.

2.3 DNS Misconfiguration and Attack Scenarios

Various misconfigurations can degrade the availability of a domain name. One problem is inconsistency between the delegation NS RRset in the parent zone and the authoritative NS RRset in the child zone. Extraneous servers referenced in the parent zone can lead to unresponsive or *lame* servers—those that don't respond authoritatively for zones that they're advertised for. Such servers, just as servers missing from the delegation RRset, might increase the chance for resolution failure.

Another common misconfiguration affecting availability is the omission of necessary glue records. This void creates a cyclic dependency, such as that shown in Figure 2, caused by a missing glue record for *ns2.foo.net*. The misconfiguration

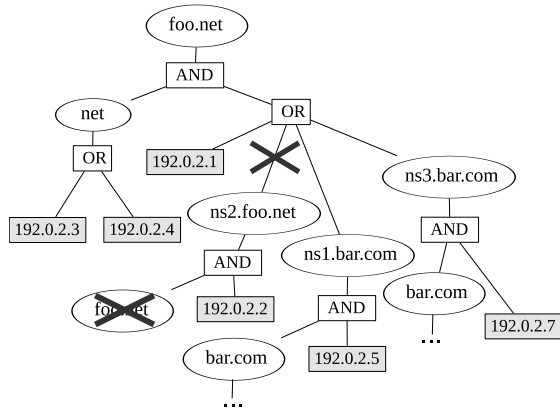


Fig. 3. A logical tree describing the availability of *foo.net*.

limits the options in the resolution path for the dependent domain name, reducing availability, as shown in Figure 3. Cyclic dependencies can also be caused by other, more subtle misconfigurations. Regardless of cause, these can result in an increased potential for resolution failure.

Proper configuration and maintenance of a DNS deployment will ensure that its namespace remains highly available for resolution. DNS misconfigurations are further explained and their impact quantified in [16, 50].

The fundamental nature of its functionality and its inherent insecurity have made the DNS the target of attack since its inception. Various vulnerabilities in protocol and implementation have facilitated exploits, which lead to compromise at higher levels. For example a malicious party might fabricate a response to a query for *www.foo.net* to redirect Web clients from its legitimate Internet address to a server set up to collect private information.

One of the largest targets in the DNS is the transport mechanism. The User Datagram Protocol (UDP) is typically used for DNS queries, and UDP identifies related packets using only source and destination port. The destination port for DNS requests (and therefore the source port for DNS responses) is well-known (port 53), so the problem space becomes guessing the unknown, 16-bit UDP port and an additional 16-bit identifier supplied in the header of the DNS packet itself.

Two prominent attacks which seek to narrow the problem space to correctly guess the bit sequence for the response that a client is expecting, are the birthday attack [76] and the Kaminsky attack [77]. In the birthday attack an attacker issues a large number of requests for the same name to a recursive resolver, resulting in an equal number of simultaneous requests open for that name on vulnerable resolver implementations. Each open request increases the chance for a successful spoof by an attacker, whose forged packet may match any of the open requests. Thus the number of attempts required by the attacker to achieve success is lessened significantly, according to the birthday paradox.

After a resolver receives a response from an authoritative server, the answer remains in its cache until it expires, which means that future queries to the resolver for the same name will not induce queries to the authoritative server. However, the Kaminsky attack skirts this limitation by making requests for non-existent names. For this approach there is no limit to queries that can be elicited by the resolver because non-existent names are in rich supply to the attacker. The objective is not to poison the RRs of the answer itself, but rather the NS RRset returned in authoritative responses, which indicates the names of authoritative servers. The attacker, if successful, can redirect all queries for names in the compromised namespace to malicious servers, now recognized by the resolver as “authoritative” for the hijacked domains. The deficiency of sufficient UDP source-port randomization in resolver implementation reduces the problem space to guessing the 16 bits from the DNS query identifier through repeated queries.

Successful injection leads to *cache poisoning*, in which an illegitimate response is stored in a resolver’s cache until expiration, which may be set arbitrarily long by the attacker. Until the response is expunged, the resolver will continue to use the false information.

3 Security Solutions for Domain Name System

In this section we describe several approaches to enhance DNS robustness. Since the DNSSEC solution can foil most DNS attacks and is being adopted by real systems, we discuss it in detail and discuss the corresponding robustness issues.

3.1 Approaches for Securing DNS

Several solutions have been proposed in the literature for securing the DNS. The foremost methodology for DNS security is the DNS Security Extensions (DNSSEC). In DNSSEC authoritative servers return cryptographic signatures with which a resolver may authenticate DNS responses using public keys. DNSSEC introduces new RR types and protocol extensions to enable this authentication. The DNSSEC protocol and its challenges will be discussed further in the next section.

DNSCurve [19] is another cryptographic solution for securing the DNS. With DNSCurve the communication channel between resolver and authoritative server is encrypted and authenticated using an elliptic-curve cryptographic system. The setup and maintenance for DNSCurve incurs very little overhead, and authoritative servers do not have any special requirements since the key is stored as the NS target of a delegation record for the domain. Authentication of DNS data with DNSCurve relies solely on the security of the channel between servers. Thus, a resolver may verify that it has received a DNS answer from an authoritative server that has not been modified in-transit. However, a third party, (e.g., a stub resolver) cannot in turn independently validate the response offline.

Several techniques have been proposed to specifically address the problem of lack of entropy, which facilitates third-party injection. Dagon, et al., [14] propose 0x20-bit encoding, which mixes the case of the owner name being queried. The authoritative server must respond with the same mixed-case query name which it received from the resolver, which increases the attacker’s problem space. WSEC DNS [54] introduces entropy by adding wildcard RRs into a zone, each of which aliases the legitimate record. Resolvers querying the authoritative servers for a name prepend a random string to the query name. The authoritative answer will respond with the correct answer because the random string will match the wildcard which aliases the real record. In addition to increasing entropy to impede would-be DNS hijackers, both of these solutions are quite backward compatible.

CoDoNS [55] is a peer-to-peer solution which proposes distributing the DNS namespace uniformly across a peer-to-peer network, as opposed to its current hierarchical structure. The hash of each domain name maps to a “home node” from which authoritative data can be queried. Name lookups are verified cryptographically using a certificate system.

DoX [82] is another peer-to-peer proposal in which the role of peers is not to host authoritative data, but rather to serve as a resource for independent “second (or n th) opinions” for name resolution. When a resolver gets an answer from a name lookup that differs from the last verified response, it issues the same query to n peers and analyzes the first m results. If the answers are entirely consistent among the peers, then the lookup is verified. If there is some disagreement among the responses, then the resolver performs its own iterative DNS name lookup, querying authoritative servers for an answer. If this answer is different from the first answer it receive, then cache poisoning is detected. Otherwise, if the agreement is less than a configured threshold, a warning is raised.

3.2 The DNSSEC Protocol

The DNS Security Extensions (DNSSEC) [3–5] add authentication to DNS, so resolvers may verify the legitimacy of a response. Public keys are included in zone data for each zone using DNSKEY RRs. Every RRset in a zone is signed by the zone’s private key(s), and each resulting signature is included in an RRSIG RR. All RRSIGs covering an RRset must be included in the response to a DNSSEC query.

Given an RRset, a covering RRSIG, and the corresponding DNSKEY, a resolver may validate the RRSIG. However, before a resolver can trust the validation, it must authenticate the DNSKEY. The digest of a DNSKEY is included in the parent zone as a DS (delegation signer) RR. The child’s DS RRset is signed by the parent zone’s keys, establishing a *secure entry point* (SEP) into the child zone. This recurrence allows a resolver to build a *chain of trust* from RRset to trust anchor at a common ancestor zone, typically the root zone. Figure 4 illustrates the chain of trust for an example DNS hierarchy. The *secure.com* zone is linked to its parent, while *island.com* is an *island of security*. Neither *broken.com* nor *insecure.com* are signed.

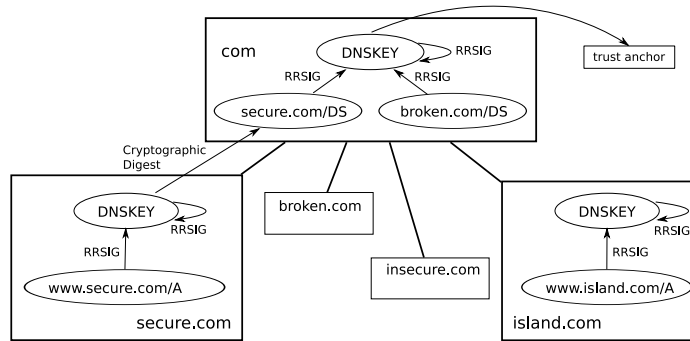


Fig. 4. The DNSSEC authentication chain for several fictitious zones. RRSIGs are represented by upward arrows extending from the RRset they cover to the DNSKEY which can validate it. SEP DNSKEYs are mapped to their corresponding trust anchor or DS RR with an arrow.

When there is no SEP from a secure parent zone to its child, the parent authoritative server must prove that no DS RRs exist, indicating an *insecure delegation*. Parent authoritative servers return signed NSEC RRs with such negative responses to demonstrate this proof. For example, a server authoritative for the *com* zone in Figure 4 should send the appropriate NSEC RR(s) in response to a query for DS RRs for the *insecure.com* and *island.com* zones.

When a resolver is configured to validate DNS responses, there are three primary outcomes with regard to validation of an RRset using DNSSEC: *secure*, *insecure*, and *bogus*. Secure responses result from an unbroken chain of trust between the RRset and a trust anchor. If the resolver can verify the secure termination of a chain of trust by way of insecure delegation, the result is an insecure response. A broken chain of trust yields a bogus response.

3.3 DNSSEC Challenges and Solutions

While DNSSEC enables authentication of RRsets, it also adds complexity to the requirements for name resolution and increases the potential for failure. Any misconfiguration of server or zone in the line of trust between anchor and name queried widens the target of error.

Among the misconfigurations contributing to validation failures are: missing, expired, or otherwise invalid RRSIGs covering zone data; missing DNSKEY RRs required to verify RRSIGs; bogus delegations caused by lack of DNSKEYs in a child zone corresponding to DS RRs in the parent zone; insufficient NSEC RRs to prove an insecure delegation to a resolver; or stale trust anchors in a resolver, which no longer match SEP DNSKEYs in the corresponding zone.

Because of the novelty of DNSSEC deployment, not all server implementations are consistent with regard to the level of DNSSEC they support. Some implementations don't even recognize the RR types (e.g., DNSKEY, DS) introduced with DNSSEC and therefore don't respond properly when queried for

them. Other implementations support DNSSEC-specific RRsets but don't follow DNSSEC protocol and don't return RRSIGs with the RRsets they cover. Finally, some DNSSEC implementations support RR types and protocol, but lack support for more recent protocol additions, such as NSEC3 for *hashed authenticated denial of existence*. These deficiencies are problematic if a signed zone is served by servers not supporting the level of DNSSEC support it requires.

Another challenge with DNSSEC is data consistency. In ordinary DNS, synchronization problems might go unnoticed for long periods of time with few ill effects. However, the time sensitivity of a signed zone requires freshness to avoid expired RRSIGs and inconsistent DNSKEY RRsets in the wake of *key rollovers*. If the master servers do not notify other authoritative servers of new zone versions, or if there are other network configurations (e.g., firewall rules) inhibiting the servers from downloading the most current version of the zone, then propagation of the zone is delayed. Even with proper notification and clear transfer paths, administrators must be sure that a newly signed zone has its *serial* number incremented after changes to signal secondary servers that they should download the new version. The `dnssec-signzone` utility distributed with the ISC BIND (version 9.6) software³ signs (or re-signs) a zone, but by default it does not increment the serial number. Improper use of this utility by administrators might lead to server inconsistency.

Several things are necessary to help administrators maintain a correct and consistent DNSSEC configuration. Tools for comprehensive analysis of a DNS deployment facilitate both understanding and troubleshooting. They should examine a deployment both hierarchically along the chain of trust and laterally across authoritative servers. DNSViz is one such tool developed to that end. DNSViz offers a graphical view of the DNSSEC chain of trust, identifies inconsistent behaviors among authoritative servers, and is available via a Web interface [21]. However, at the moment, DNSViz lacks an application programming interface (API), which would allow administrators to monitor their DNS programmatically. Other online tools exist for regular deployment tracking [68, 47, 28], monitoring [75] and real-time analysis [20, 18, 84]. Zones should be routinely monitored and thoroughly analyzed for correct behavior.

Maintaining correct behavior is a matter for the owners of the authoritative DNS service of a zone. However, problems caused by inconsistent responses may also be mitigated by the validating resolver. If validation of an RRset fails because of bogus or incomplete validation data received from an authoritative server, some implementations re-issue the query to other authoritative servers, attempting to complete the chain of trust. Such is the case with the ISC BIND resolver.

Even if a resolver practices such validation diligence, inconsistent response behavior results in reduced redundancy of hosted zones and more traffic to those servers serving legitimate responses. Additionally, if the resolver is behind one or more *proxy* resolvers to which it is configured to forward its requests, it is at the whim of the proxy resolvers, regardless of whether or not the upstream

³ <http://www.isc.org/software/bind>

resolvers are configured for validation or have practiced validation diligence to obtain appropriate DNSSEC responses.

Another method by which resolvers might mitigate misconfiguration in the chain of trust is by caching authenticated DNSKEYs for use as trust anchors at lower levels in the chain of trust [17]. If a resolver maintains such a *soft anchor*, using methodology similar to that in RFC 5011 [6], it can continue to authenticate RRsets below a misconfigured ancestor zone where the chain of trust is otherwise broken. Vantages [78] is another project that makes DNSKEYs available as trust anchors through alternate means. In Vantages DNSKEYs found to be consistent as viewed by different peers are shared among a peer-to-peer network known as a community of trust.

DNS is an essential component of the Internet's architecture. DNSSEC deployment is under way to protect its integrity, but the additional complexity of DNSSEC challenges the availability of DNS. Various DNSSEC-related server and zone misconfigurations can affect not only the corresponding zones, but also the entire namespace below.

4 Secure End-to-End Communication Protocols

Secure end-to-end communication is required to ensure authenticity and privacy of vital data, and can be provided at multiple levels with different objectives. In the following, we discuss two popular protocols, namely IPsec for network level security and SSL/TLS for session level security. With increasing popularity of metro-Ethernet [42], it is necessary to provide comprehensive security at Ethernet level as well; however, currently there is no such protocol available (although encrypted transfer has been introduced by several vendors for metro-Ethernet).

4.1 Network Layer Security

The most popular means of providing network layer security is via the IPsec protocol that provides IP level confidentiality, integrity and authentication. IPsec can effectively address the common problem of IP address spoofing and man-in-the-middle attacks. The IP address authentication required for this also prevents anonymous denial-of-service attacks. The standards define several new packet formats: the authentication header (AH) to provide data integrity and the encapsulating security payload (ESP) to provide confidentiality and data integrity. The Internet Key Exchange (IKE) is used for Key management and for negotiating the security parameters between the devices.

IPsec provides two modes of operation: *transport mode* and *tunnel mode*. In the transport mode, only the IP payload is encrypted, and the original IP headers are left intact. This mode has the advantage of adding only a few bytes to each packet. It also allows devices on the public network to see the final source and destination of the packet, and thus IP QoS can be easily implemented without any changes. However, since the transport layer header will be encrypted, deeper packet inspection becomes impossible (including even the determination whether

the packet belongs to a TCP flow or another transport protocol). Also, since the IP header remains intact, it is possible to do IP level traffic analysis.

In tunnel mode, the entire original IP datagram is encrypted and encapsulated in a new IP packet. This mode allows a network device, such as the source gateway router, to act as an IPsec proxy. That is, the source router performs encryption on behalf of the hosts and forwards the packets using normal Internet routing towards the destination IPsec proxy router. The destination router decrypts the original IP datagram and forwards it on to the destination system. The major advantage of tunnel mode is that the end systems do not need to be modified to make use of IPsec. Tunnel mode also protects against traffic analysis since only the source and destination router addresses (i.e. the tunnel endpoints) are revealed. However, unless the tunneled packets are specially marked, it is not possible to provide any special QoS to them.

IPsec works by creating security association (SA) at each end. The security association stores the negotiated parameters of the association and is referred to by a randomly chosen number called security parameter index (SPI) and the destination IP address. IPsec operation assumes the existence of SA, which itself is set up via the Internet Key Management (IKE) protocol. IKE creates an authenticated, secure tunnel between two entities and then negotiates the security association for IPsec. This process requires that the two entities authenticate themselves to each other and establish shared keys. IKE supports multiple authentication methods including pre-shared keys, public key infrastructure (PKI) based authentication, and PKI based digital signatures. Party A can authenticate party B by sending a nonce encrypted with B's public key, which B decrypts, encrypts it with A's public key and sends back to A. If A can correctly recover its original nonce, then it must be communicating with the "true" party B. Authentication is usually needed only at the beginning of a session or occasionally. The two parties can make their communication undeniable by signing the message (or a hash over the message) with its private key. The shared key can be established using a separate Diffie-Hellman exchange or via the use of PKI.

These two steps—authentication and key exchange—create the IKE SA, a secure tunnel between the two devices. One side of the tunnel offers a set of algorithms, and the other side must then accept one of the offers or reject the entire connection. When the two sides have agreed on which algorithms to use, they must derive key material to use for IPsec with authentication header (AH), encapsulating security protocol (ESP), or both together. IPsec uses a different shared key than IKE. The IPsec shared key can be derived by using Diffie-Hellman again to ensure perfect forward secrecy, or by refreshing the shared secret derived from the original Diffie-Hellman exchange that generated the IKE SA by hashing it with pseudo-random numbers (nonces). The first method provides greater security but is slower. After this is complete, the IPsec SA is established.

A common use of IPsec is in defining a Virtual Private Network (VPN), where multiple segments of a private corporate network are linked over a public network using encrypted tunnels. This allows applications on the private network

to communicate securely without any local cryptographic support, since the VPN routers perform the encryption and decryption. IPSEC is well suited for this environment, more so than tunneling PPP (point to point protocol) over SSL or SSH, since it operates directly on the IP packets and preserves a one-to-one correspondence between packets inside and outside the network. This is particularly important for the streaming media traffic such as voice over IP (VoIP). In contrast, a PPP based tunnel over an encrypted TCP connection would experience a substantial stall whenever TCP experiences a packet loss (due to built in TCP mechanisms).

We now discuss some of the issues with IPsec. As with any security protocol, a ubiquitous deployment of IPsec in the Internet would imply substantial complexity in key management. The same applies regarding the overhead of cryptographic operations, particularly the authentication and shared key establishment; however, this can be addressed using hardware implementations. Network Address Translation (NAT) is a popular mechanism to address IPv4 address limitations. NAT works by replacing the destination address in a packet by a local address. When NAT changes the IP addresses or ports in the IP header, IPsec cannot re-calculate the hash because it does not know the key and so IPsec drops the packets. In ESP the NAT device cannot access and change the port information inside the encrypted TCP headers of the packets. The normal workaround for this is to provide an additional UDP header that can be updated.

4.2 Session Level Security

At the session level, security is often provided by SSL (secure sockets layer), also referred to as TLS (transport layer security). The use of TLS is controlled by individual application; therefore, unlike IPsec, only those communications that use TLS are protected. TLS also cannot prevent IP level attacks. On the other hand, it provides better granularity than IPsec in that the authentication and encryption can be selected as needed by the application and the documents to be exchanged.

Individual applications (e.g., email, remote access) may use other secure communication mechanisms depending on their needs. For example, PGP (pretty good privacy) is quite popular for email, and Kerberos and SSH are commonly used for remote access. XML security mechanisms support even finer level of security via SAML (security association markup language) whereby different parts of a document can be protected differently. For example, sensitive parts of the documents can be encrypted selectively and the access rights (e.g., read, update, append, ...) can also be specified at that granularity.

SSL involves two important functions: (a) authentication of the communicating parties via the *SSL handshake protocol*, and (b) encryption/decryption of data exchanged between the two parties during the session. The handshake protocol uses the public key infrastructure (PKI) and establishes a shared symmetric keys between the parties to ensure confidentiality and integrity of the communicated data. The handshake involves three phases, with one or more messages exchanged between client and server:

1. Phase 1 starts with client sending information regarding the algorithms and parameters that it can handle. This includes shared key exchange algorithm (e.g., RSA, Diffie-Hellman, etc.), bulk encryption algorithm to be used (e.g., AES, IDEA, ...), and message authentication algorithm (SHA1, SHA-256). Each of these algorithms have their own parameters (e.g., key length) that also must be communicated.
2. Phase 2 involves mutual authentication of client and server by exchanging their credentials. The server authentication is always based on a PKI digital certificate that server must possess and supply to the client; however, the client rarely supplies its certificate; instead the client authentication is typically done following the handshake via a login/password supplied to the server.
3. Phase 3 involves the client sending the shared key to the server. This is encrypted with servers public key so that only the addressed server is able to decipher it.

The server must obtain its PKI certificate from a trusted certification authority (CA). The cost and scalability considerations naturally lead to a hierarchical certification model. For example, the organization providing the web service may receive its certificate from a globally known CA such as VeriSign or GoDaddy in exchange for a fee. This certificate will be signed using the private key of the global CA. The organization could then issue certificates for the servers it owns, signed by its private key that it obtained from the global CA. When a client receives a server certificate, it would need the organization's certificate to decode it, which it must receive it as well. The organization's certificate is signed by the global CA, which the client's browser would typically configure statically. Thus, so long as the global CA's can be trusted, the chain of authentication can authenticate any lower level entity as well.

Following the handshake protocol, the data exchange takes place using the chosen private key bulk-data encryption algorithm. Although numerous encryption algorithms abound, the development of a US standard called Advanced Encryption Standard (AES) has made it a popular algorithm. AES divides the input message into 128-bit xed blocks and encrypts each block into ciphertext with a 128, 192, or 256-bit key. The encryption algorithm consists of 10, 12, or 14 rounds of transformations depending on the key size. Each round uses a different round key generated from the original key using a key schedule. In the cipher-block chaining (CBC) mode of AES, each plaintext block is XORed with the ciphertext block of the previous block (along with a especially chosen initial block). SSL also decomposes large messages into fragments of size at most 16 KB in order to facilitate message authentication. Each fragment is appended with the message authentication code (MAC) which is computed as follows; a copy of the fragment is first appended with the secret key and message sequence number, and then whole thing is reduced to a fixed size MAC using the negotiated secure hash function (SHA1, SHA-256).

The use of SSL can result in a significant slow down due to four factors [32]:
(a) Three round-trip network delays during handshake, (b) additional network

traffic and hence higher queuing delays resulting from handshake, (c) significant computational expense of PKI, and (d) throughput degradation associated with bulk data encryption. The true performance impact depends on how much data is exchanged per session, round-trip network delays, and available bandwidth. A careful implementation of bulk encryption algorithms can often run with only a minor difference between encrypted and unencrypted data stream. This is particularly true when the hardware necessary features such as fast rotates. On the other hand, PKI involves a considerable amount of overhead. RSA is by far the most commonly used public key encryption algorithm in practice and belongs to the class of exponentiation ciphers [31]. Let (e, d) denote the encryption/decryption key-pair, and $N = pq$ where p and q are large primes. Then, encrypted message C and plain-text message M are related as follows:

$$C = M^e \bmod N, \quad M = C^d \bmod N \quad (1)$$

Here the public key consists of the pair (N, e) and the private key consists of the triplet (p, q, d) . The strength of this cipher is essentially controlled by the difficulty of factoring N into the factors p and q , which necessitates key lengths of 1024 or 2048 bits. The algorithm to efficiently evaluate equation (1) involves unsigned arithmetic operations (multiplication, addition, shift and rotate) on large integers. Thus, processors with longer word length unsigned integer instructions will automatically provide a significant performance boost to RSA. In fact, since a $2n \times 2n$ -bit multiplication requires four $n \times n$ -bit multiplications, doubling the word length of unsigned operations will provide 4-times performance boost so long the cycles per instruction remains the same. It turns out that even with a 1024-bit key length, almost 80% of the SSL handshake time may be spent in the RSA exponentiation, which makes the efficient coding of exponentiation critical for good performance. SSL acceleration has been studied widely in the past and highly optimized software implementations, acceleration using special instructions and features, and outright hardware implementation of SSL are widely available [32, 48, 57]. Lately several papers have examined acceleration of PKI using GPUs—one such recent work is [27].

SSL has been adopted widely for securing web based services. Although SSL can indeed provide strong security if used properly, its deployment suffers from several real-world problems. First, if the key length chosen is too small, the cipher can be broken via a variety of known attack techniques. In particular, older implementations may use PKI key length of 512 bits, which is inadequate; the current recommendation being 2048 bits. Similarly, key lengths of less than 128 for bulk encryption could be problematic. A similar issue arises with respect to the choice of the algorithms as well. The MD5 secure hashing algorithm was shown to suffer from the collision problem in 2005, and constructions that can actually generate a fake certificate with the same MD5 hash as a real certificate were shown in 2008 [71]. Yet the use of MD5 has not completely disappeared. A related problem has been the diligence on the part of certificate authorities (CA's) such as VeriSign or GoDaddy in ensuring that the certificates do not violate any of the standard assumptions about them. For example, certificates only make sense for fully qualified domain names (FQDNs), but there have been many

reports of certificates being issued for fragments of FDQNs, which can be mis-used (see <https://www.eff.org/press/mentions/2011/4/7-1>). Furthermore, since the certificates issued by CAs are often pre-installed by the browsers, a compromised browser could contain unverified CA's or fake CA's could be installed later in the browser.

Although the indication of secure access by including https in the URL (instead of http) made for an easy implementation, such an approach forces the user to know whether to use HTTP or HTTPS. This can provide for an easy attack path since most users don't know or don't care for the distinction and simply click to continue on when warned of missing, expired, invalid or self-signed certificates. Furthermore, if the legitimate organization (e.g., a bank) provides only HTTPS access to its site, it is easy for an attacker to set up a corresponding HTTP site and trick the user into giving the password via DNS poisoning, directing DNS queries to a malicious DNS server, or simply by luring the user to use a wireless hot-spot under attacker's control. A better integration of security that does not require user to do anything different can instead provide better security.

5 Integrity of Internet Routing

Routing is a fundamental and essential function of Internet infrastructure and its integrity is paramount. Routing integrity could be violated due to misconfigurations, policy conflicts, and attacks. These issues become more difficult to handle as the sophistication and complexity of networking increases. In this section we provide an overview of routing integrity issues and discuss several proposed solutions.

The Internet architecture employs interconnection between many *Autonomous Systems* or (AS). An Autonomous System is characterized by a single management authority that provides consistent routing within its own *domain* and packet exchanges with other domains. The routing algorithms used within an AS are referred to as Interior Gateway Protocols (IGP) or intra-domain routing protocol, and popular choices are OSPF (Open shortest path first) and iBGP (interior border gateway protocol). The routing protocol between ASes (or inter-domain routing protocol) is typically known as Exterior Gateway protocols, and the EBGp (exterior BGP) or simply BGP has become the de-facto standard for it.

Intra-domain routing protocols tend to be simpler and their configuration is controlled by a single entity. This makes routing misconfigurations less likely and easier to detect. Disruption of intradomain routing is also less valuable from an attackers perspective since the impact is likely to be confined to a single domain. For this reason, security issues are generally less critical for intra-domain routing than inter-domain routing.

5.1 Security and Robustness of BGP

BGP [56] belongs to the class of *path vector* routing protocols, wherein each node advertises the “best” route for each destination to all of its neighbors. A BGP node stores all the paths sent by its neighbors but uses and advertises only the one that is “best” according to some given policy. When this primary path fails, BGP withdraws this path and selects the next best backup route. The new route is then advertised to its neighbors.

The fundamental reason for BGP’s popularity is its great flexibility in setting up and using routes so that each ISP can implement its desired policies without having to make them public. Unfortunately, this flexibility and opacity is also the cause of many problems with BGP including incompatible configurations by different ISPs, lack of global visibility into the quality or status of routes, and inability to mount a coordinated approach to handling routing problems [45, 36]. For example, even if all paths are picked strictly based on cost estimates, BGP often shows undesirable behavior such as long convergence delays and oscillations because the path availability information is typically deduced rather than explicitly propagated. As a result, BGP may substitute a failed path with the next best one without realizing that its local assessment of this path is incorrect [51]. This absence of information about the validity of a route can cause BGP to go through a number of backup routes before selecting a valid one. The cycle of withdraws/advertisements can continue for a considerable amount of time and this delay is known as the *convergence delay* (or *recovery time*). These vulnerabilities can even be exploited by an attacker to make BGP behave in a worse way than otherwise.

The convergence problem has been examined extensively in the literature [24, 35, 36, 45]. In particular, it was shown by Labovitz et al. [35] that the convergence delay for isolated route withdrawals can be greater than 3 minutes in 30% of the cases and could be as high as 15 minutes. They also found that packet loss rate can increase by 30x and packet delay by 4x during recovery. There are also many attempts to improve BGP convergence delay via a variety of techniques [10, 52]. Reference [66] examines the issue of packet delivery performance and proposes a technique to reduce packet loss and delay.

We now examine the robustness of the routing tables themselves. Although it is difficult to corrupt routing tables directly, legitimate routing table update messages could be hijacked in order to create inconsistent routing tables or bogus messages update messages could be sent simply exhaust computing, memory or other resources at the routers. A related issue is that of perturbation to the QoS settings, which can be exploited to disrupt proper traffic treatment. A suppression, duplication, or change to route update messages can cause misdelivery and congestion. Such attacks have been considered extensively in the literature, and the protection is via cryptographic means such as authentication of update messages and encryption of packet contents or headers. Reference [11] provides a comprehensive survey of BGP vulnerabilities and numerous protection mechanisms.

Secure BGP (SBGP) protocol [34] uses signed route updates by making use of two PKI hierarchies, one to ensure integrity of IP prefixes and the other for AS numbers. The main goals of Secure BGP is to ensure integrity of “AS-path” recorded in route updates and the advertised IP prefixes. (AS-path is the list of ASes through which a BGP route advertisement or withdrawal message travels before reaching its destination.) The use of PKI hierarchy is essential to have a chain of trust for every IP prefix and AS path; however, the need for multiple public key operations makes the scheme quite slow. Hu, Perrig and Johnson [25] discusses a fast mechanism for securing routing updates for BGP. The authors make a case for using symmetric cyptography, which is much more efficient than using PKI (public key cryptography infrastructure). They advocate the use of hash chains involving message authentication code (MAC) at each node along the path taken by update packets to ensure that the updates are not tampered with and nodes are added to or removed from the path. Although such a scheme can secure updates, there are still issues of key exchange and the overhead of checking MACs across hash chains. Interdomain Route Validation (IRV) is a very different authentication where an IRV server in each AS can optionally validate the update from the originator of the AS.

Although cryptographic techniques are able to secure routing updates against unauthorized changes, they bring in extra complexity and thus increase chances of misconfigurations. In fact, the problems of SBGP, which uses a PKI hierarchy, are similar to the DNSSEC issues discussed in section 3.2.

In rare cases, an AS may itself turn malicious perhaps due to compromise by an adversary. For example a malicious BGP AS may decide not to withdraw a nonworking route, or fail to put a working route into the set of current routes. A malicious AS could also falsely advertise that it has the most desirable (e.g., shortest) path to certain destination. This will cause all traffic to the destination be routed through this AS and the AS may either snoop on the traffic or simply discard part or all of it and cause denial of service. The modification to AS-path can also cause similar impacts. Achieving robust routing in spite of some malicious ASes can be very difficult, but it may suffice to simply detect and quarantine such ASes.

5.2 Behavior of BGP under Large Scale Failures

Large-scale failures can be caused by a number of reasons such as malicious electronic attacks on the Internet infrastructure, earthquakes, major power outages, hurricanes, terrorist activities, etc. Increasingly, communication networks are needed the most during times of crisis, and that increases the importance of a quick recovery. Thus it is vital that we have a good understanding of BGP convergence behavior after large-scale failures. The primary reason why large scale failures in the Internet have not been studied is their low probability of occurrence and the complexity in analyzing them. However, the recently there has been an unmistakable trend toward much more frequent and extreme events that can disrupt Internet substantially. Detailed results on the impact of large

scale failures on BGP are contained in [62, 64, 65, 63, 66] and are summarized here.

A large-scale failure typically spans multiple ASes, in either a contiguous or dispersed geographical area. Besides significantly degrading the connectivity from and to the affected ASes, large scale failures will also have a big impact on the connectivity between the source-destination pairs that use the affected ASes for transit. It has been observed that the routes to the most popular prefixes/ASes in the Internet are remarkably stable [59], presumably because those ASes are well designed and maintained. However a large scale failure far away from these popular prefixes/ASes can potentially tear down routes to these ASes from large parts of the Internet.

In studying the impact of large-scale BGP router failures on BGP performance depends on a number of parameters including:

1. Magnitude of failure, measured in terms of number of AS'es affected, or more precisely the number of external BGP (EBGP) routers that have failed.
2. Topological characteristics of the failure, e.g., failure confined to one big cluster of routers vs. scattered throughout the entire network.
3. Distance based preference in inter-AS connectivity, i.e., the extent to which relative geographic location of the EBGP routers affects connectivity. (This, along with the last aspect covers the impact of geographic distribution on the performance.)
4. Degree distribution for inter-AS connectivity (average degree and its variability).
5. Value of the MRAI (Minimum Route Advertisement Interval), which is the minimum interval between route advertisements for the same prefix.

These situations are analyzed in detail in [62] and only a few cases are explicitly discussed here. The results were obtained via detailed simulation of BGP using SSFNet [40] and the network topology was generated by BRITe [39]. In order to examine the impact of failure size on BGP convergence delay in a simple setting, we considered a network with constant degree. We used the degree distribution derived from the Internet AS-level topology but decided to restrict the maximum degree to 40 for our 120 AS experimental network. This gave us a degree distribution which decays as a power law with an exponent of about -1.9. The average degree is about 3.67. Fig 5 compares the convergence delays for three degree distributions with average degree set to 3.67. The degree distributions chosen are: (a) constant (actually 3 or 4 with an average of 3.67), (b) A mixture of 70% low degree (1-3) and 30% high degree (7-8), and (c) "Realistic" degree, i.e., one based on our measurements (i.e., power law with exponent of -1.9).

The plotted recovery time is in seconds and is given as a function of the failure magnitude (in terms of fraction of routers failed). It is seen that the recovery time increases initially with the size of the failure to some maximum value and then slowly rolls off. The recovery time rises initially because, a larger failure translates into more failed routes and more failed backup routes. However as the number of failed nodes continues to grow, the residual network gets smaller and

hence the length of the backup routes explored during the convergence process is shortened. This causes the decline in the convergence delay. The curves for the three degree distributions show that the recovery time is the highest for constant degree and lowest for the “realistic” case. This, somewhat surprising result comes from the fact that in a power-law type of degree distribution the unaffected high degree nodes provide connectivity to a large number of nodes and thus help BGP recover faster.

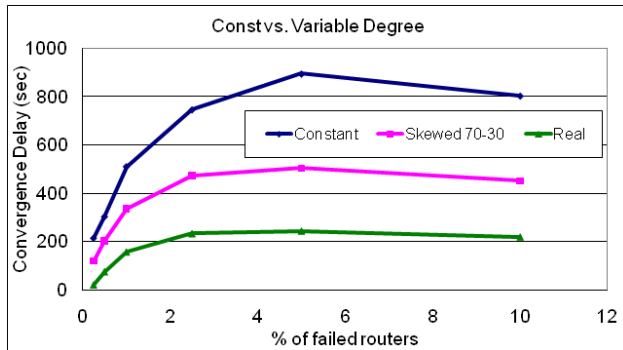


Fig. 5. Recovery time for different degree distributions

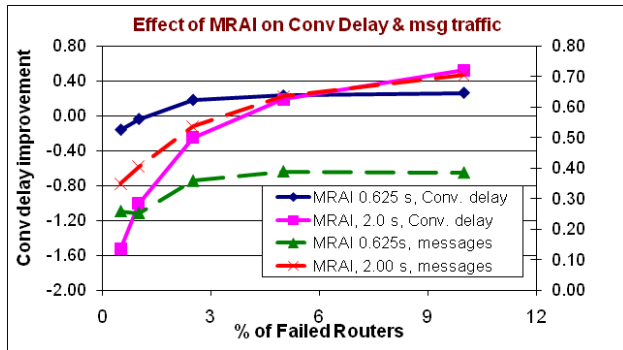


Fig. 6. Convergence delay for different sized failures

We next show the impact of MRAI value on the convergence delay for various sized failures as shown in Fig 6. We restricted the size of the failures to 10% because larger failures are probably not realistic and may take down too many routers to be interesting. The figure shows the fractional change with respect to the baseline case for which the MRAI was set to 0.25secs. From the results we can see that a low MRAI results in a low convergence delay for small failures,

but the delay increases rapidly as the size of the failure goes up. For higher MRAI values, the convergence delay for small failures is greater (than that for low MRAI values), but the convergence delay for large failures is lower. The figure also shows the fractional change for the number of generated messages for the three different MRAI values. The trend is similar to that for the convergence delays. For small failures, the number of messages is low and about the same for all the MRAI values.

For brevity, we do not include detailed results for a number of other cases that were studied. However, the study does point to the following characteristics:

- The convergence delay initially increases and then goes down as the size of the failure is increased.
- Higher average inter-AS degree leads to higher convergence delays.
- A small number of well connected nodes in a topology reduces the convergence delay (as compared to a case where all ASes have constant degree)
- The convergence delay first decreases and then increases as the MRAI is increased (First observed by Griffin and Premore [24]). The “ideal MRAI” for a failure increases with the size of the failure.
- The ideal MRAI for a particular topology depends on the degree of the best connected ASes.
- The convergence delay increases with the number of ASes in the topology.
- The convergence delay is reduced, if ASes preferentially connect to nearby ASes.

In order to reduce the convergence delays for large scale failure, several schemes have been investigated in [65] that do not require any coordination between ASes or any changes to the protocols. The dynamic MRAI scheme automatically tries to select the “optimal” MRAI for a failure, based on the size of the message queue at the router. The scheme works very well and the convergence delay is always close to the minimum for failures of various magnitudes. The dynamic scheme reduces the convergence delays for large scale failures while keeping the delays low for smaller, more probable failures. The parameters for this scheme are the three different MRAI values and the two thresholds, all selected based on experimental results. A new batching scheme was also examined reduces the generation of invalid route advertisements and removes stale update messages during periods of overload. Such a batching scheme can substantially cut down the convergence delays. The convergence delays were reduced even further if we combined the batching and the dynamic MRAI schemes. Another advantage of the batching scheme is that it does not use any configuration parameters.

It has been found that the convergence delays is not necessarily strongly correlated with the packet delivery performance, and the latter is really what matters from a user’s perspective. Reference [66] therefore proposes techniques to reduce packet loss and delay for large scale failures. Fig. 7 shows improvement in packet loss rate as compared to the baseline case for 4 different schemes. The consistency assertion (CA) [52] and speculative invalidation (SI) [64] schemes were designed for reducing convergence delays and are included unchanged. We also included modified versions of these, developed in [66], where the modifications

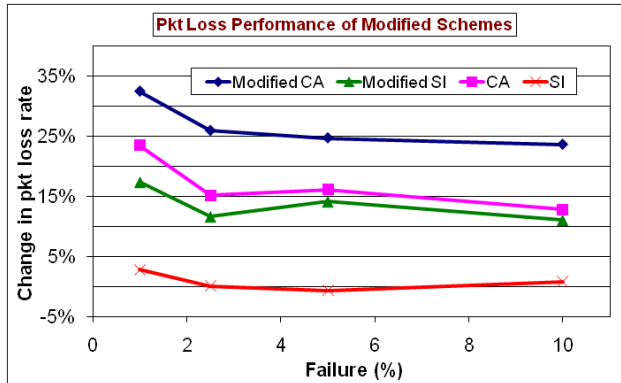


Fig. 7. Packet loss for Various Schemes

are directed towards reducing the packet delays. It is seen that the modifications improve the performance of the schemes considerably.

The packet delays during the recovery depend on the number of update messages which in turn depends on the number of destinations affected by the failure. Despite the advances in router processor speeds, a large scale failure in the Internet, which contains nearly 200,000 destinations, will generate a huge number of updates that is likely to overwhelm a large number of routers. Thus suppression of redundant messages is essential for keeping packet delays under control.

5.3 Routing Misconfiguration

Misconfiguration of routing tables is a different and frequently occurring problem that is not amenable to cryptographic means. A common misconfiguration is the advertisement of a route for a prefix which is not entirely served by the router. This essentially creates a “black-hole” for packets destined to addresses that belong to the advertised prefix, but are not served by the router and hence discarded. Similarly, improperly selected alternate routes can lead to packet loss and route convergence issues in case of failure or management withdrawal of the primary route. Finally, incorrect packet discrimination rules may discard desired packets or accept unwanted ones. These issues are often addressed by checking route configurations against the rules and policies expressed declaratively. Some recent research in area concerns discovering the rules and policies by data mining instead of being pre-specified [37]. Reference [8] considers a similar mining approach for access control misconfigurations. Access-control policy misconfigurations can cause requests to be erroneously denied, and mining of access history is used to determine consistency between access control intent and policies.

Routers invariably support many services beyond simple routing and integrity of such services and their configuration data must be maintained. Consider, for

example, MPLS (Multi-Protocol Label Switching) which allows creation of a virtual network in IP backbone by encapsulating packets using a MPLS header [60]. The encapsulation header is used to route packets along Label Switched Paths (LSPs). A label is a short, fixed-length identifier that is associated at the ingress router and ensures that all packets with the same label will follow the same LSP to the destination or the egress router, without regard to the original IP packet header information. The Label Distribution Protocol (LDP) is used to create labels and bind them to LSPs. The MPLS labels are not secured and thus could be targets of attack. Reference [49] discusses such attacks and proposes a framework for hardening the MPLS network routing by requiring digital signature of all the signaling messages for the MPLS control plane protocols. Reference [2] proposes multipath routing with a (k, n) threshold secret sharing to secure MPLS packet contents. Such a mechanism requires compromise of at least k segments of the packet (out of n) in order to be able to reconstruct the original packet.

The most popular service provided by routers is the support of quality of service (QoS) either on a hop-by-hop basis (e.g., differentiated services, or Diff-Serv) or via end-to-end reservation of resources, as in IntServ or RSVP. DiffServ packets must be marked at the originating end-point or ingress router with the correct DiffServ Code Point (DSCP) which is then used to provide appropriate treatment in terms of priority scheduling and traffic shaping. Thus, improper marking by malicious sources or due to misconfiguration of marking logic can easily disrupt other traffic [72]. Both of these can be very difficult to detect until they cause disruption. RSVP involves end to end signaling to reserve resources before a communication takes place. Reference [73] proposes mechanisms to secure RSVP based on cryptographic authentication and encryption have been proposed. In all cases, however, the routers involved must store the necessary configuration parameters for the QoS treatment, and integrity of this data needs to be ensured. Furthermore, security only increases the complexity of the configuration and increases chances of misconfiguration.

Distributed Denial of service (DDoS) attacks are a common ailment in networks and can be launched via botnets distributed throughout the network and targeting the victim end nodes, routers, or entire network segments. DDoS attacks can be difficult to track down if each botnet sends only a small amount of traffic and each of them attempts to hide their identity. Reference [53] provides a survey of such attacks and mechanisms for countering them.

6 Integrity of Layer 2 Switching

A comprehensive network integrity protection demands appropriate consideration of all networking layers—PHY, MAC, IP, Transport and application. The PHY/MAC attacks (and hence integrity) can be either technology dependent (e.g., jamming of or snooping on wireless link) or technology independent (broken link or MAC address spoofing). In this chapter we do not focus on technology dependent issues except as they concern configuration management issues.

Although layer-3 routing is most visible in the Internet context, layer-2 switching is becoming important because of metro-area MAC technologies such as metro-Ethernet, WiMax or LTE. In these instances, layer-2 domains could cover a large area and hence layer-2 attacks or misconfiguration could have a wide-spread impact. Even at a local level, data centers invariably use layer-2 devices (switches) because of their much lower cost, latency, and configuration complexity as compared with routers. In fact, even a large data center with 10-100K servers may use only a few routers at the periphery. Thus security of layer-2 switching becomes critical.

VLANs (Virtual local area networks) allow network administrators to divide layer-2 networks into a set of smaller logical networks. Each VLAN consists of a single broadcast domain that is isolated from other VLANs. The simplest way to set up VLANs is by assigning switch ports to appropriate VLAN numbers. Thus a port belongs to a specified VLAN and an incoming packet can only be directed to the ports assigned to the same VLAN. In more sophisticated versions, a port may belong to multiple VLANs, or a VLAN may be indicated via the subnet that the port belongs to. The packets themselves can be tagged with VLAN id using IEEE 802.1Q standard that allows 12-bits for VLAN id (in addition to 3 bits of priority and 2 bytes of tag protocol ID). An incoming tagged packet is accepted only if the packet VLAN id matches with one of the VLAN ids associated with the port. VLANs also allow transfer of untagged packets but the port must indicate how it addresses the untagged packet.

VLANs can easily extend across switches and create multiple layer-2 networks. Cisco provides a proprietary protocol called virtual trunking protocol (VTP) to automatically propagate VLANs across switches. (In the absence of such a protocol, VLANs would have to be set manually on each switch.) The spanning tree protocol (STP) in this case needs to be VLAN aware so that packets can be routed properly from one switch to another along appropriate ports. It is also possible to transfer packets between switch along “trunks”, i.e., along a connection between switch that allows packets with any VLAN id to pass through to the other side. Cisco has a proprietary protocol called DTP (dynamic trunking protocol) that can automatically set up trunks between switches and thus simplifies setup.

The switch configuration data includes a number of attributes including VLAN setup, size of address table and number of MAC addresses per port, L2 QoS setup (assuming data center Ethernet capabilities), interface speeds and corresponding parameters, etc. A number of attacks on switches have been identified, many of them having to do with VLANs [61].

In order to route packets among switches, each switch needs a routing table—implemented via a content addressable memory (CAM)—that lists the destination MAC address and the corresponding switch port to which the incoming packet should be directed to. Unlike IP level routing algorithms, inter-switch routing does not follow an elaborate path-setup procedure. Instead, if a packet comes in with a new destination MAC address, the switch simply broadcasts it to all ports (i.e., the switch acts like a hub). The receiving port will either dis-

card the packet or forward it to the next switch, if any. The connection between switches is governed by one of the many variations for the basic spanning tree protocol (STP), which ensures that for every VLAN, the paths from any given switch to all other ports form a tree. Thus, a packet can eventually make it to its destination along a unique path. The response to the broadcast indicates to the source port the correct destination port which is then deposited in the routing table. The CAM entries usually have an associated timeout period to ensure that a faulty path does not stay forever.

This mechanism can be misused by simply flooding a switch with packets with new destination MAC addresses so that all CAM entries will fill up and the switch will start broadcasting. This would cause congestion and excessive delays for all packets, including new legitimate packets that will be unable to establish a table entry. More important, by strategically flooding the desired switches, the attacker can force legitimate packets to be routed to ports that it has access to and sniff them there. This problem can be addressed by limiting the number of MAC addresses that a given port can learn in a given time period.

As switches become more complex, so do the chances of misconfiguration and ability to exploit various “loopholes”. For example, the automatic setting up of trunks using DTP can result in a problem if the port states on two sides of a link are not set up properly. In particular, trunking may be enabled when it really was not intended, or an end device can connect to a switch and pretend to be a switch by participating in the DTP protocol and thereby get access to the packets from other switches. In this case, the attacker will be able to get hold of packets belonging to any VLAN. The protections against these attacks consist of enabling only those ports and features that are really needed. For example, DTP should be enabled only on ports that are intended to be trunk ports (Cisco default enables DTP on all ports). Unfortunately, such recommendations are the primary cause of configuration errors in the first place, since they refer to manual actions that may be forgotten or misapplied.

In order to enable preservation of VLAN information end-to-end, the IEEE 802.1Q extension allows VLAN information to be encapsulated into another 802.1Q frame. This feature effectively allows “VLAN hopping”, i.e., forwarding of a packet from one VLAN (used for encapsulation) to another (inner) VLAN. When such a packet arrives at the first switch, the outer VLAN tag is stripped and the packet is sent to the VLAN specified by the inner 802.1Q tag. A hacker can exploit this to route packets from other VLANs to his own VLAN ports where he can sniff them. A slight variation of this idea is to have the inner tag specify a trunk VLAN. Then the first switch will broadcast the packet to all ports (after stripping the outer tag). Thus if the hacker has access to any of the ports, he can sniff the traffic. Such an attack can be made harder by using dedicated VLAN IDs for all trunk ports.

The identification of the spanning tree in a switch network is one via a protocol that involves the so-called bridge PDUs (BPDUs). The BPDUs protocol selects a root bridge based on the configured switch priorities (or other information such as MAC addresses). It then finds a least cost path from every switch to

the root switch and thereby builds the spanning tree. An attacker can perturb this process by pretending its attacking system to be a switch and exploiting the BPDU protocol to make itself the root. It can then examine all the packets. Although there are some proprietary solutions to this problem, there is currently no general solution.

A very simple attack concerns the misuse of ARP packets. An ARP request is typically broadcast by a host asking for the MAC address corresponding to an IP address. Normally, the correct entity will respond with its MAC address, which is then cached in the ARP cache. An attacker can respond by returning its own MAC address as the ARP response. On the IP side, the attacker can spoof the IP address of an existing but inactive host, so that the response from the attacker is the only response received in response to the ARP request. The attacker's MAC address could then be inserted into the switch tables as discussed earlier. This attack is also difficult to foil. Partial solutions include restricting MAC addresses for ports (not very practical) and timeouts on ARP cache values.

7 Configuration Management Security

Nearly all systems, services and software deployed in the cyberinfrastructure involve a large number of configuration parameters which must be properly set and managed. For example, a device such as a NIC or IO controller has many settable parameters that can be altered in firmware and may affect the behavior either immediately or as a result of some action such as a reboot. As the sophistication of the services and security mechanisms to protect them increases, so does the complexity of configuration management. This leads to increasing vulnerability resulting from misconfigurations and from attacks designed to perturb the configuration. Configuration errors have been estimated to enable 65% of cyber attacks and cause 62% of infrastructure downtime. This could be due to configuration conflicts, inconsistency, bugs that lead to incorrect implementation of security configuration. We have already discussed the configuration issues with DNS and DNSSEC and their consequences. Routing algorithms such as BGP have similar issues and several substantial incidents have already occurred in the past. For example, routing misconfiguration that directed all YouTube traffic to Pakistan Telecom in Feb 2008, route flooding of Worldcom routers in 2002, etc. A well known problem with BGP route setup is the aggregation of prefixes into a larger prefix along with "holes" that need to be advertised separately. A failure to properly address all parts of the address space leads to packets being misrouted or ending up in black-hole.

Detection of misconfigurations for a given network protocol or system requires a deep understanding and is often very difficult. Nevertheless, with systematic configuration management systems becoming prevalent, it is important to examine issues in a general setting and address some of the more generic problems. For example, redundant configuration information can be both a source of inconsistencies and an opportunity to detect problems. In this chapter we discuss

configuration management issues in a generic way and discuss some mechanisms to harden it.

7.1 Storage of Configuration Data

These configuration parameters could be stored in a variety of ways depending on the entity involved. For example, low level configuration data such as features of a NIC or hardware configuration of a server are invariably maintained in a firmware repository, whereas OS and higher level data is maintained in “configuration files” or software databases. Configuration files are very difficult to manage and are usually protected only via access control mechanisms (e.g., read by all, update by “root” only). There is an ongoing trend towards using XML based databases for configuration management data, as it allows for convenient and automated management. For example, although traditionally virtual machine (VM) configuration has been specified via files, the open virtualization format [46] standardizes it by exploiting the CIM (common information model) standard [13]. This enables the VM configuration to be stored in CIM repositories (much like the lower level data stored in firmware CIM repositories). A recent virtualization management initiative called VMAN [79] attempts to provide standardized mechanisms to manage virtual machines stored in CIM repositories in OVF format. Data centers already use CIM based repositories extensively all the way from CIM based firmware repositories for individual devices all the way up to management of clusters of servers allocated for a given application.

CIM models allow a compact representation of the configuration by using concepts like instantiation, inheritance, and aggregation. CIM models can be accessed in multiple ways, including the popular web services based management (WSMAN) that runs atop SOAP (simple object access protocol), which itself is layered on top of HTTP. Although this web-services based management is very convenient, it exposes configuration to web-services based attacks as discussed below. WSMAN does support authentication and encryption, but they are often bypassed due to performance reasons, thereby opening the door for web-services based attacks.

7.2 Characteristics of Configuration Data Corruption

Much of the configuration data is typically used only occasionally or under special circumstances. For example, most of the hardware settings are scanned only at machine boot time and many of VM parameters reloaded only at VM reset time. The consequence of this infrequent reading of configuration parameters is that a corruption could lay dormant for a long time, and when it does manifest itself, it may be impossible trace back its source. Thus protection and frequent consistency checking of configuration parameters becomes very important.

The ability to corrupt some CIM data on one asset (server, switch, etc.) or devices (e.g., NICs, disk controllers, etc.) usually means that the same vulnerability exists on a large number of other entities of the same type and those too

could be afflicted similarly. This is the result of limited diversity: data centers typically like to deploy identical asset for ease of installation and management. Consequently, once an attacker finds a way to corrupt the CM data on one asset, it can easily turn this exploit into a large scale concerted attack. However, the redundancy can also be exploited for checking the consistency of the configuration as discussed below.

Similar issues arises with respect to duplicate storage of duplicate information by the management system. For example, device configuration information may be read from firmware repositories at boot time and stored in software databases during normal operation for quick access. Thus a hacker could corrupt this more easily accessible data in SW databases even if access to lower level repositories is difficult or protected by cryptographic or access control mechanisms. On the positive side, if such a corruption can be detected, correct values can be restored from the device. A special case of duplication is aggregated information (e.g., count of number of servers with certain characteristics). In this case, an attack on the aggregated data can cause wide-spread impact, but with an effective detection mechanism, it is easy to reconstruct the data correctly.

7.3 Web Services Based Attacks

Web-services based management is becoming very popular due to the availability of a wide variety of software development tools. Unfortunately, this also makes configuration repositories vulnerable to a host of attacks on web services [70]. As with other web-services, WSMAN interfaces for manipulating configuration data are described using WSDL (Web Services Definition Language) and the descriptions are often automatically generated. Automatic generation often means that descriptions include *all* supported procedures including those not really intended for use by non-developers. Hackers can exploit these as an easy mechanism to corrupt configuration data. Similarly, publication of the web services via a public directory such as UDDI simplifies the job of the hacker.

Since SOAP headers in WSMAN commands use XML and require XML parsing, it is possible to craft bogus headers that nevertheless require significant parsing effort. Also, a SAX (Simple API for XML) based parser (that extracts all relevant information in a single pass) can be easily tricked into overwriting earlier values. This is one mechanism by which a legitimate update to configuration variable can be hijacked to produce an invalid or otherwise problematic value. Another mechanism concerns the misuse of *XML external entities*, which is merely a macro facility by which one could include contents of external files in the XML stream. If the hacker can overwrite or replace such a file, it can put arbitrary XML code there. One such possibility is to open a new TCP connection with the privileges of the XML parser and perform arbitrary data transfer. A related attack is XML schema poisoning to alter control flow or otherwise cause incorrect processing of XML data. Finally, the hacker can inject arbitrary data wrapped in XML (e.g., XPATH expressions, SQL queries, LDAP requests, etc.) to achieve specific attacks related to how the configuration data is manipulated.

Some of the XML manipulation attacks can disable authentication and thereby gain unrestricted access to the configuration data.

7.4 Protecting Data Center Network Configuration

In this section we discuss a simple mechanism for protecting the configuration management of data center networks by exploiting the considerable redundancy that exists naturally. Data center networks typically follow the “fat-tree” architecture shown in Fig. 8. Here a node at level ℓ is a level- ℓ switch, and m such switches may be used to provide the desired bandwidth and reliability. Often $m = 2$ for rack or chassis level switches, but more redundancy may be used at higher levels. The m switches are typically identical not only in their make and model number, but also in terms of their detailed configuration. The only exceptions are some unique parameters such as the MAC address, port numbers, or the fact that they are connected to different switches on the other end. We first describe the mechanism by considering only the redundant data—non-unique and “similar” data cases are addressed later.

The configuration data of the low level devices is stored in firmware repository with each device, and there are redundant copies of some of these data in the management servers. Even if the management servers are trusted, they control the devices using WSMAN, and thus an attacker may launch web services based attacks to manipulate the configurations of the devices and the compromised data will also reflect in the management servers too.

Fig. 8 shows additional infrastructure required to protect the configuration data. At the top level, we have a global management server (MS) for the entire data center, and at each lower level, we have a local management server. At each level of the fat tree, one node is designated as “reference” and is marked as (R) in Fig. 8. The reference node is considered as the *gold standard* and its configuration data is strongly protected. The reference nodes are controlled by the corresponding management servers. Other nodes of the same type verify their data against this gold standard.

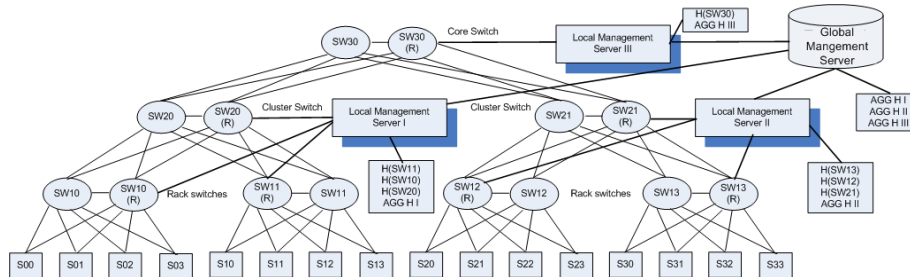


Fig. 8. Fat Tree Structure of Data Center Network

The details of the algorithm are given in [33] and are only summarized here. During the initialization, reference nodes are chosen (random choice is fine because of period rotation) and they bind strongly with the MS'es. The integrity checking is based on secure hash of the configuration data, henceforth denoted as $H(SW_n)$ for switch n . The hashes are stored at the management side. The aggregated hash values of the local management component are also stored at the global MS; these are indicated as AGG tuples. All updates to the reference nodes are secured and allowed only by authenticated entities. After the reference node successfully updates its configuration, it recomputes a new hash value and sends it back to the MS. It also updates the non-reference nodes.

The configuration data in non-reference nodes is verified against the gold standard, but the time and frequency of verification depends on the tradeoff between performance and security. One possible approach is to allow use of data without verification but do verification on each update request. An alternate method is to do the verification periodically. It may also be useful to do a periodic verification even if updates are verified. Upon detection of a misconfiguration or attack, the correct data can be loaded from the reference node. The data in the reference node itself may need periodic verification, and this can be done by a normal discovery process that each node undertakes upon startup. The reference nodes themselves can be rotated periodically (at a much lower frequency) so as to foil reconnaissance attempts by hackers directed towards the reference node. The newly chosen reference node must be first authenticated and verified by the local MS before the current reference node is dislodged. This handoff can be easily handled as described in [33]. The rotation is a form of *moving target defense* mechanism in action [69], and it ensures that even if an insecure link is compromised, it will be detected after some time.

Notice that the secure update is issued from the local MS; consequently, the server side will get the response and keep a log entry of the update. If a malicious intruder pretends to be a local MS and updates the configuration, there will be no record of the update on the real MS side, and malicious updates can be easily detected. Attacks trying to defeat the protection by overriding the address of the reference node or bypassing the check can also be detected. It is because that when the TTL expires and a verification is invoked, the reference node can not receive hash values from non-reference nodes, and as discussed above, the links binding reference nodes and the local MS can hardly be compromised. In addition, legitimate and malicious modifications to the configuration data can be distinguished by the channels. Modifications using the secure update via local MS are considered legitimate, and all other modifications can be treated as malicious.

In the above, we assumed—for simplicity—that all configuration data is identical for members of the set. In practice, each node will always have some unique data as discussed above. Any updates to such data must be authenticated, much like the data maintained by the reference node. It is also possible to allow small variations in the configuration parameter values in a variety of ways including simple trimming the actual value down to a lower bound for hash calculation.

A more sophisticated approach involves use of locality sensitive hashing (LSH), which ensures that two parameter values that are close result in a hash values that are also close [41]. For effective use of LSH, the parameters can be divided into groups according to the level of specificity of information. It is also possible to divide the parameters into groups and use the mechanism for each group separately. It is likely that the appropriate technique will depend on the type of data being protected and the consequences of perturbations to it, and such issues are currently largely unexplored.

8 Conclusions and Future Challenges

In this chapter we discussed several aspects of providing security and robustness in the Internet. In particular, we focused on the robustness of fundamental functions of name resolution, routing, and switching. We discussed some protocols such as IPsec and SSL for ensuring secure transfer of information. Finally, we discussed the issue of robustness and security of configuration management since many of the challenges in maintaining robustness can be attributed to protocol and configuration complexities.

It is a well accepted fact that the original design of Internet protocols did not consider security issues at all, and security was added later in a rather haphazard manner. Although many of the well known Internet protocols have been either standardized or floated as a draft standard, only a small fraction of them have either come into widespread use (e.g., SSL) or have received enough traction (e.g., DNSSEC, IPsec). Often the problem is the embedded base of unsecured protocols and the overhead of implementing secure versions. There are still many protocols that are notoriously insecure (e.g., ARP, DHCP, RSTP, ICMP, etc.) but almost universally deployed without much protections. Practical schemes for securing such protocols remains a challenging problem. Finally, there are many proprietary protocols that are in operation (e.g., VTP/DTP mentioned in section 6) or proprietary ways of securing standard protocols (e.g., root guard mechanism for STP). Standardization of such mechanisms remains a major challenge.

The recent push towards widespread DNSSEC deployment has been driven by the extreme vulnerability of the plain DNS protocol and the routine phishing and pharming attacks that it allows. Although, this is a welcome development, the significantly higher configuration complexity of DNSSEC can significantly impact the name resolution availability as detailed in section 3.2. Automated troubleshooting of DNSSEC configurations still remains a challenging problem. Furthermore, the DNSSEC experience has important lessons for securing other protocols: a careful consideration of potential misconfigurations and inconsistencies, design decisions to avoid or at least minimize them, and functionality to mitigate their effects.

Given the deficiencies of current Internet in several areas including security, support for mobile and cloud computing and scalability, the research community has been working on “clean slate” Internet architectures. Recently, four major

projects [29] on “future internet architecture” (FIA) have started to take shape. All of these projects intend to build security into the architecture from ground up to avoid many of the problems of current internet. However, this emphasis on security and other features (e.g., support for mobility and virtualization) are likely to result in much more complex architectures, where configuration errors may be difficult to detect and handle. The major challenge is to address configuration issues also and design in mechanisms in the architecture to detect and react to configuration related vulnerabilities. As is borne out from the current Internet, subtle misconfigurations could significantly degrade a variety of vital functionalities such as name resolution, switching, routing, firewall protection, etc. irrespective of the precise architecture used.

There are many outstanding challenges in understanding, detection, avoidance, and mitigation of misconfigurations. Some of the more generic challenges are detailed in [30]. One very poorly understood aspect of misconfigurations is the relationship between misconfiguration and malfunctioning. This translates into numerous issues such as misconfiguration dependencies, characterization of events that trigger a misconfiguration (e.g., reboot, software reset, etc.), configuration dependencies that lead to misconfiguration of one component resulting in misbehavior of another, classification of misconfigurations in terms of their ill effects, etc. As a simple example, a single configuration is often used for multiple entities (e.g., single “configuration file” for several Virtual Machines). In other cases, a default configuration may have been set initially but remains unchanged. The practice of keeping aggregate data also creates a strong dependency in that the corruption of the aggregate value affects proper usage of the corresponding assets. There are many other situations dependencies but have not been explored in the literature. Finally, mitigation methods to address misconfigurations without violating the standards or requiring significant changes to implementation is a very challenging issue that usually cannot be dealt with in a generic way.

9 Exercises

1. What is the purpose for glue records, when are they required, and what happens to the availability of a DNS zone when required glue records missing?
2. Given the following DNS zones: (a) root zone - signed, (b) com zone - signed, with DS RRs existing, and (c) example.com zone - unsigned, and a validating resolver that is anchored at the root zone, what behaviors might cause the validation failure of a name in the example.com zone? What configuration, compatibility, or administrative problems might cause these behaviors?
3. Compare and contrast the primary cryptographic methods for securing DNS (DNSSEC and DNSCurve) with the non-cryptographic approaches described in this chapter, including the pros and cons of each.

Acknowledgements: The authors would like to thank the collaborators of our research which forms the basis of much of the coverage in this chapters. In particular, we would like to thank Prasant Mohapatra (UC/Davis), Jeff Sedayao

(Intel), Amit Sahoo (Cisco), Lihua Yuan (Microsoft), Meixing Le (GMU), and Ravi Iyer (Intel).

References

1. M. Al-Fares, A. Loukissas, A. Vahdat, “A scalable commodity data center network architecture”, Proc. of ACM SIGCOMM, Aug 2008.
2. S. Alouneh, A. En-Nouaary and A. Agrawal, “MPLS security: an approach for unicast and multicast environments”, Annals of Telecommunications, Vol 64, 2009, pp391-400.
3. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “DNS Security Introduction and Requirements”, RFC 4033, Mar. 2005.
4. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “Resource Records for the DNS Security Extensions”, RFC 4034, Mar. 2005.
5. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “Protocol Modifications for the DNS Security Extensions”, RFC 4035, Mar. 2005.
6. M. StJohns, “Automated Updates of DNS Security (DNSSEC) Trust Anchors”, RFC 5011, Sep. 2007.
7. H. Ballani and P. Francis, “CONMan: taking the complexity out of network management”, Proc. of ACM SIGCOMM Workshop on Internet Network Management, Sept 2006, pp41-46
8. L. Bauer, S. Garriss, M.K. Reiter, “Detecting and resolving policy misconfigurations in access-control systems”, In Proc. of 13th ACM Symposium on Access Control Models and Technologies, June 2008, pp185-194.
9. S. Berger, R. Cceres, D. Pendarakis, et al., “TVDC: managing security in the trusted virtual datacenter”, SIGOPS Oper. Syst. Rev. 42, 1 (Jan. 2008), pp 40-47.
10. A. Bremner-Barr, Y. Afek, and S. Schwarz, “Improved BGP convergence via ghost ushing”, in Proc. IEEE INFOCOM 2003, vol. 2, San Francisco, CA, Mar. 30Apr. 3, 2003, pp. 927937.
11. K. Butler, T. Farley, T. McDaniel, J. Rexford, “A Survey of BGP Security Issues and Solutions”, Proc. of IEEE, Jan 2010, pp100-122.
12. S. Cabuk, C.I. Dalton, H. Ramasamy, M. Schunter, “Towards automated provisioning of secure virtualized networks”, Proc. of 14th ACM CCS conference, Oct 2007, pp 235-245.
13. “Common Information Model”, Available at www.wbemsolutions.com/tutorials/CIM/cim-specification.html
14. D. Dagon, M. Antonakakis, P. Vixie, T. Jinmei, and Wenke Lee, “Increased DNS Forgery Resistance Through 0x20-bit Encoding: Security Via Leet Queries”, Proc. of 15th ACM CCS Conference, Oct 2008.
15. C. Deccio, C.C. Chen, J. Sedayao, K. Kant, and P. Mohapatra, “Quality of Name Resolution in the Domain Name System”, in *Proc. ICNP 2009*, Princeton, NJ, Oct. 13–16, 2009, pp. 113–122.
16. C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra, “Measuring Availability in the Domain Name System”, in Proc of INFOCOM, San Diego, CA, March 2010.
17. C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra, “Quantifying and Improving DNSSEC Availability”, in *Proc. ICCCN 2011*, Maui, HI, Jul. 13 – Aug. 4, 2011.
18. DNSCheck by .SE. Available at <http://dnscheck.iis.se/>
19. “DNSCurve: Usable security for DNS”, Available at <http://dnscurve.org/>

20. DNSSEC Debugger by VeriSign Labs. Available at <http://dnssec-debugger.verisignlabs.com/>
21. “DNSViz”, Available at <http://dnsviz.net/>
22. T. Garfinkel and M. Rosenblum, “When Virtual Is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments”, USENIX Association, 2005
23. P. Goyal, R. Mikkilineni, M. Ganti, “FCAPS in the business services fabric management”, Proc. of 18th IEEE Intl. workshop on Enabling Technologies, 2009.
24. T.G. Griffin and B.J. Premore, “An experimental analysis of BGP convergence time”, in *Proc. ICNP 2001*, Riverside, California, Nov. 11–14, 2001, pp. 53–61.
25. Y-C. Hu, A. Perrig and D.B. Johnson, “SPV: Secure Path Vector Routing for Securing BGP”, Proc. of SIGCOMM, Aug 2004,
26. Intel Active Management Technology. Available at en.wikipedia.org/wiki/Intel_Active_Management_Technology
27. K. Jang, S. Han, S. Han, et.al, SSLShader: Cheap SSL Acceleration with Commodity Processors, <http://www.ndsl.kaist.edu/papers/sslshader.pdf>, Proc. of NSDI 2011.
28. IKS-Jena. Available at <https://www.iks-jena.de/leistungen/dnssec.php>
29. J. Pan, S. Paul, R. Jain, “A survey of the research on future internet architectures,” IEEE Communications Magazine, IEEE , vol.49, no.7, pp.26-36, July 2011
30. K. Kant, “Configuration Management Security in Data Center Environments”, Invited chapter in *Moving Target Defense: Creating Asymmetric Uncertainty for Cyberthreats*, Eds. S. Jajodia, A. Ghosh, V. Swarup, C. Wang, X. Wang (to appear).
31. J. Katz and Y. Lindell, *Introduction to Modern Cryptography: Principles and Protocols*, Chapman & Hall, 2007.
32. K. Kant, R. Iyer and P. Mohapatra, “Architectural Impact of Secure Socket Layer on Internet Servers”, International Conference on Computer Design (ICCD 2000), Sept 2000, pp 7-14.
33. K. Kant and M. Le, “Security Considerations in Data Center Configuration Management”, available at www.kkant.net/download.html
34. S. Kent, C. Lynn, and K. Seo, “Secure Border Gateway Protocol (S-BGP)”, IEEE Journal on Selected Areas in Communications, 18(4):582592, April 2000.
35. C. Labovitz, A. Ahuja, et al., “Delayed internet routing convergence”, in *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, Aug. 28–Sep. 1, 2000, pp. 175–187.
36. C. Labovitz, A. Ahuja, et al., “The Impact of Internet Policy and Topology on Delayed Routing Convergence”, in *Proc. IEEE INFOCOM 2001*, vol. 1, Anchorage, Alaska, Apr. 22–26, 2001, pp. 537–546.
37. F. Le, S. Lee, T. Wong, et. al, “Detecting network-wide and router-specific misconfigurations through data mining”, IEEE/ACM Trans. on networking, vol 17, No 1, Feb 2009, pp 66-79.
38. C. E. Leiserson, “Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing”, IEEE Trans. on Computers, Vol 34, No 10, pp892901, 1985.
39. A. Medina, A. Lakhina, et al., “Brite: Universal topology generation from a user’s perspective”, in *Proc. MASCOTS 2001*, Cincinnati, Ohio, August 15–18, 2001, pp. 346-353.
40. “SSFNet: Scalable Simulation Framework”. [Online]. Available: <http://www.ssfnet.org/>
41. M. Slaney, M. Casey, “Locality-Sensitive Hashing for Finding Nearest Neighbors”, Signal Processing Magazine, IEEE , vol.25, no.2, pp.128-131, March 2008
42. G. Chiruvolu, “Issues and Approaches on Extending Ethernet Beyond LANs”, IEEE Communications Magazine, vol. 42, no. 3, , 2004.

43. P. Mockapetris, "Domain Names - Concepts and Facilities", RFC 1034, Nov. 1987.
44. P. Mockapetris, "Domain Names - Implementation and Specification", RFC 1035, Nov. 1987.
45. D. Obradovic, "Real-time Model and Convergence Time of BGP", in *Proc. IEEE INFOCOM 2002*, vol. 2, New York, Jun. 23–27, 2002, pp. 893–901.
46. "Open Virtualization Format", Available at dmtf.org/sites/default/files/standards/documents/DSP2021_1.0.0.tar
47. E. Osterweil, D. Massey, and L. Zhang, "Deploying and monitoring DNS security (DNSSEC)", in *Proc. ACSAC '09*, December 2009.
48. D. A. Osvik, J. W. Bos, D. Stefan, and D. Canright. *Fast software AES encryption*, In Foundations of Software Engineering (FSE), 2010.
49. F. Palmieri and U. Fiore, "Enhanced security strategies for MPLS signaling", *Journal of Networks*, Vol 2, No. 5, Sept 2007.
50. V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang, "Impact of configuration errors on DNS robustness", in *Proc. ACM SIGCOMM 2004*, Portland, OR, Aug. 30–Sep. 3, 2004.
51. D. Pei and B. Zhang et al., "An analysis of convergence delay in path vector routing protocols", *Computer Networks* 30 (3) (2006), pp. 398421
52. D. Pei, X. Zhao, et al., "Improving BGP convergence through consistency assertions", in *Proc. IEEE INFOCOM 2002*, vol. 2, New York, NY, June 2327, 2002, pp. 902911.
53. T. Peng, C. Leckie, K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems", *ACM Comput. Surv.* 39, 1 (Apr. 2007).
54. R. Perdisci, M. Antonakakis, X. Luo, and W. Lee, "WSEC DNS: Protecting Recursive DNS Resolvers from Poisoning Attack", *Proc. of dependable systems and networks (DSN)*, June 2009.
55. V. Ramasubramanian and E.G. Sirer, "The design and implementation of a next generation name service for the Internet", in *Proc. ACM SIGCOMM 2004*, Portland, OR, Aug. 30–Sep. 3, 2004.
56. Y. Rekhter, T. Li, and S. Hares "Border Gateway Protocol 4", RFC 4271, Jan. 2006.
57. E. Rescorla, A. Cain, and B. Korver. SSLACC: A Clustered SSL Accelerator. In *USENIX Security Symposium*, 2002.
58. J.S. Reuben. A Survey on Virtual Machine Security. Helsinki University of Technology, 2007. Available at http://www.tml.tkk.fi/Publications/C/25/papers/Reuben_final.pdf
59. J. Rexford, J. Wang, et al., "BGP routing stability of popular destinations", in *Proc. Internet Measurement Workshop 2002*, Marseille, France, Nov. 6–8, 2002, pp. 197–202.
60. [1] E. Rosen, A. Viswanathan, and R. Callon, "Multi-protocol Label Switching Architecture", IETF, RFC 3031, 2001.
61. S.A. Rouiller, "Virtual LAN security: weaknesses and countermeasures", available at uploads.askapache.com/2006/12/vlan-security-3.pdf
62. A. Sahoo, K. Kant, and P. Mohapatra, "BGP Convergence Delay under Large-Scale Failures: Characterization and Solutions", *Computer Communications*, Vol 32, No 7, May 2009, pp1207-1218.
63. A. Sahoo, K. Kant, and P. Mohapatra, "Characterization of BGP recovery time under massive internet failures", *Proc. of Intl conference on communications (ICC)*, Istanbul, June 2006.

64. A. Sahoo, K. Kant, P. Mohapatra, Speculative Route Invalidation to Improve Performance of BGP under Large Scale Failures, Proc. of ICCCN 2006, Oct 2006, Washington DC.
65. A. Sahoo, K. Kant, and P. Mohapatra, "Improving BGP convergence delay for large scale failures", Proc. of dependable systems and networks (DSN), June 2006.
66. A. Sahoo, K. Kant, P. Mohapatra, Improving Packet Delivery Performance of BGP During Large-Scale Failures", Proc of Globecom 2007, Washington DC, Nov 2007.
67. R. Sailer, T. Jaeger, E. Valdez, et al, "Building a MAC-based Security Architecture for the Xen Opensource Hypervisor", 21st Annual Computer Security Applications Conference (ACSAC), Dec 2005.
68. SecSpider. Available at <http://secspider.cs.ucla.edu/>
69. F.T. Sheldon and C. Vishik, "Moving toward trustworthy systems: R&D Essentials", IEEE Computer magazine, Sept 2010, pp 31-40.
70. A. Stamos and S. Stender, "Attacking Web Services: The Next Generation of Vulnerable Enterprise Applications", Proc. of Defcon XIII. Available at www.isecpartners.com/.../iSEC-Attacking-Web-Services.DefCon.pdf.
71. M. Stevens, A. Sotirov, J. Appelbaum, et al, "Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate", Lecture notes in computer science, Vol 5677, 2009.
72. A. Striegel, "Security Issues in a Differentiated Services Internet", Proc. of HiPC workshop, 2002.
73. V. Talwar, K. Nahrstedt, S.K. Nath, "RSVP-SQOS : A SECURE RSVP PROTOCOL", Proc. of IEEE Intl. conf. on Multimedia and Expo (ICME'01), 2001
74. R. Teixeira, S. Agarwal, and J. Rexford, "BGP routing changes: Merging views from two ISPs", *ACM SIGCOMM Computer Communications Review*, vol. 35, issue 5, pp. 79-82, Oct. 2005.
75. TLDmon by DNS-OARC. Available at <https://www.dns-oarc.net/oarc/services/tldmon>
76. US-CERT, "Vulnerability Note VU#457875: Various DNS service implementations generate multiple simultaneous queries for the same resource record", Nov 2002.
77. US-CERT, "Vulnerability Note VU#800113: Multiple DNS implementations vulnerable to cache poisoning", July 2008.
78. Vantages, Available at <http://www.vantage-points.org/>
79. "Virtualization Management (VMAN) Initiative : DMTF Standards for Virtualization Management", Available at <http://www.dmtf.org/standards/vman>
80. Web service security specification, available at docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
81. Web services secure conversation specification, available at specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf
82. L. Yuan, K. Kant, P. Mohapatra and C.N. Chuah, "DoX: Domain name cross checking: An antidote for DNS cache poisoning", Proc. of Intl conference on communications (ICC), Istanbul, June 2006.
83. L. Yuan, K. Kant, P. Mohapatra, C.N. Chuah, "A proxy view of quality of domain name service", Proc of INFOCOM, Anchorage, AL, May 2007.
84. ZoneCheck by AFNIC. Available at <http://www.zonecheck.fr/>