

SoK: An Analysis of Protocol Design: Avoiding Traps for Implementation and Deployment

Tobias Fiebig*, Franziska Lichtblau*, Florian Streibelt*,
Thorben Krüger*, Pieter Lexis†,
Randy Bush‡, Anja Feldmann*
*TU Berlin

{tobias, franziska, florian, thorben, anja}@inet.tu-berlin.de

†PowerDNS.COM BV

pieter.lexis@powerdns.com

‡Internet Initiative Japan

randy@psg.com

Abstract

Today's Internet utilizes a multitude of different protocols. While some of these protocols were first implemented and used and later documented, other were first specified and then implemented. Regardless of how protocols came to be, their definitions can contain traps that lead to insecure implementations or deployments. A classical example is insufficiently strict authentication requirements in a protocol specification. The resulting Misconfigurations, i.e., not enabling strong authentication, are common root causes for Internet security incidents. Indeed, Internet protocols have been commonly designed without security in mind which leads to a multitude of misconfiguration traps. While this is slowly changing, to strict security considerations can have a similarly bad effect. Due to complex implementations and insufficient documentation, security features may remain unused, leaving deployments vulnerable.

In this paper we provide a systematization of the security traps found in common Internet protocols. By separating protocols in four classes we identify major factors that lead to common security traps. These insights together with observations about end-user centric usability and security by default are then used to derive recommendations for improving existing and designing new protocols—without such security sensitive traps for operators, implementors and users.

1. Introduction

Security incidents involving Internet services have become regular events. Examples include: (a) disclosures of information, e.g., petabytes of personal data stored in unprotected key-value stores - NoSQL databases [119], [18]. (b) Unauthorized access via the Internet to systems, e.g., supervisory control and data acquisition systems (SCADA) [116] which are used to control systems which range from light installations to oil platforms. (c) Undesired publication of information, e.g., health data from the United Kingdom on an HTTP root [159], or an UEFI signing key on an internal FTP server which was available via the Internet [41].

While programming vulnerabilities are well-studied, e.g., see [114], [113], insufficient attention has been paid to poorly designed and hard to configure protocols. We focus on protocol security practices with their varying naïvete, complexity, and weaknesses. In the context of this paper, we use the term protocol to refer to transport and application layer communication protocols. We use the term service to refer to a deployed instance that is offering the service associated with the protocol.

Indeed, among the root causes of the above severe incidents are misconfigurations. While this is, in principle, well known, e.g., [53], [177], to date, the main and only explanation has been human error. We claim that there are more fundamental reasons for such misconfigurations which stem from the design of the Internet's protocols themselves, the security assumptions or configuration choices offered by protocols. These lead to services which are prone to

misconfiguration.

Our study complements the multitude of individual incidents documented in the scientific literature and the anecdotes in systems lore with a macroscopic systematic survey of Internet protocols and their corresponding services. More precisely, we investigate which of the underlying assumptions during protocol design lead to misconfigurations during service deployment. We refer to this as misconfiguration prone protocols/services.

In the context of this paper we consider the following security relevant misconfigurations. A service is subject to misconfiguration if it is deployed on the Internet in such a way that any of the three main security properties - confidentiality, integrity, or availability (CIA) - can be tainted.

There are many reasons for misconfigurations: (a) the operator does not follow best practices regarding network settings by which the service is deployed, (b) the operator does not use the default configuration settings leading to tainted CIA, or (c) the operator uses the default configuration settings and they lead to tainted CIA.

Given the newest Internet trends, services in the cloud, Internet of Things including Industry 4.0, autonomous systems, e.g., self-driving cars, and mobile applications, we expect even more diversity and complexity and, thus, more security incidents. Considering the link between mismanagement and maliciousness [183], [107], many of these incidents will involve misconfigurations. Thus, we claim that a systematic review of why misconfiguration occurs is needed.

Contributions: We review the assumptions under which protocols have been designed along two dimensions, (a) the assumed strength of the attacker - weak vs. strong - and (b) the defense paradigm - good enough vs. perfect security. By using these dimensions to group protocols we find four major clusters, one in each quadrant. We name these clusters *Early Internet* for weak attacker/good enough, *Emerging Threats* for weak attacker/perfect security, *Complex Security* for strong attacker/perfect security, and *A new Simplicity* for strong attacker/good enough. The names capture the design essentials as well as the mindset of the protocols in each of the classes. Furthermore, the names express how the security mindset of protocol design evolved over time. From this systematization of misconfiguration prone protocols we derive a set of specific action items. To keep protocols secure and misconfiguration resilient these must be considered when introducing new or updating old protocol specifications.

Structure: This paper is structured as follows: We

introduce our systematization methods in Section 2. Next, we dedicate a chapter to each of the four classes, Sections 3 to 6, examining how and why protocols and services in the specific class are prone to misconfiguration. For each class we show its properties based on a few examples which we discuss in detail. Each of these chapters concludes with a discussion which summarizes our observations about the misconfigurations within the class. Finally, in Section 7, we provide design recommendations for future protocols as well as deployed services, and end with concluding remarks.

2. Systematization Method

Thousands of Internet protocols have been proposed, developed, and deployed on the Internet; therefore an exhaustive analysis is beyond the scope of this paper. Rather we extract essential features from a subset of misconfiguration prone protocols and use these to derive our systematization.

2.1. Example protocol selection

Our choice of protocols is driven by the following considerations. First, we choose protocols that are commonly used and/or are new and upcoming. Next, we choose some where misconfigurations can potentially have a large impact or those where system lore states that they are easily misconfigured. We augment this list by protocols that capture corner cases. From these we selected a set of protocols that are most iconic for the relevant class. Due to the size limitations of the work at hand, only a subset can be introduced in detail and displayed in Table 2. We focus on the server, not client side. Thus, pure end-user focused protocols as well as client misconfigurations are beyond the scope of this paper.

2.2. Security relevant misconfigurations

If a service is deployed in such a way that its CIA is tainted because of a misconfiguration we call the misconfiguration security relevant. There are three main reasons for such misconfigurations. The first is when the operator of the service does not follow best common practice (BCP). The second is when the operator uses their own configuration which taints the CIA of the service. The third is when the operator uses the default configuration, but the CIA is still tainted, likely due to incorrect defaults.

We refer to misconfiguration prone protocols and services. But in the end it is the service that is misconfigured, often facilitated by design choices in

the protocol. Examples for common misconfigurations are that the service is deployed in a network setting which deviates from the one for which it, or its default configuration, was designed.

2.3. Security guidelines for protocol design

Request For Comments (RFCs) document Internet protocols and services. Since 1992, each RFC must address the topic of security, according to, IETF processes, (RFC1311 [135], RFC1543 [136], RFC2223 [137], RFC7322 [69]), which document what an RFC must contain. Indeed, RFC1311 [135] states: “*All STD RFCs must contain a section that discusses the security considerations of the procedures that are the main topic of the RFC.*”

Over time, the community has realized that this statement by itself is not sufficient and the specification of the security requirements have gotten stricter, see RFC3552 [144] from 2003, the Best Current Practices for “*Writing RFC Text on Security Considerations*”. The goal of this requirement is to make all protocol designers and implementers aware of possible security implications. Given that this basic requirement existed in 1992, we conclude that the importance of security has been recognized for at least 23 years.

2.4. Review of security threats for protocols and services

One of the motivations for including a security section in each RFC is to make the protocol designer consider the following two questions: (a) against whom to defend and (b) how to defend.

We find that protocol designers consider different kinds of attackers, ranging from very weak to very strong. The weak attacker is either unskilled or is resource limited. The strong attacker is very skilled and has all necessary resources in their hands.

Defining how to defend is more difficult as it depends on the eyes of the beholder. Some argue that the cost of breaking security should be larger than the value of the protected asset. This goes back to Pfleeger and Pfleeger [132] and specifies that one should put up a wall against threats at least high enough that most attacks will not break the wall. Moreover, the wall should not cost more than the protected asset. We call this the “good enough” approach to security. Others, especially the field of cryptography [102], follow the approach of “perfect security”. Perfect security refers to using every possible mean to achieve security. We refer to these two approaches as the defense paradigm.

2.5. Classification of protocols and services

Thus, we have two-dimensions, namely, the capabilities of the attacker and the defense paradigm. Using these dimensions we classify our selected set of protocols and identify four major clusters. These clusters correspond to the four quadrants of the two dimensional space. We refer to them as: *Early Internet*, *Emerging Threats*, *Complex Security*, *A new Simplicity*.

Weak attacker - good enough

This class contains those protocols that are designed for a friendly, collaborative environment - the Internet when security was not yet a major concern. This class, in particular, contains those protocols that initially were designed under the assumptions that there is no attacker, or still carry artifacts from that idyllic time. Such an attacker is the weakest one possible. We refer to this class as *Early Internet*.

Weak attacker - perfect security

This class no longer assumes that the environment is entirely friendly. Rather it recognizes that there are threats, but not yet by sophisticated attackers. However, since significant assets can be at stake, even attacks that are only theoretically possible are considered. We refer to this class as *Emerging Threats*.

Strong attacker - perfect security

This class captures the protocols that consider security a necessity at all costs. As a result, the protocols in this class are designed to handle strong attackers and be safe against all, even theoretically conceived, attack vectors. However, as a result, they are often complex and hard to deploy, maintain, and difficult to use. We refer to this class as *Complex Security*.

Strong attacker - good enough

This class contains those protocols who’s designers recognize that strong attackers exist but also value protocols that “just work” out of the box. Therefore, the designer does not try to defend the asset against every possible attack by reducing the attack surface. In this class we see a conscious choice between security and operational ease, favoring the latter. We refer to this class as *A new Simplicity*.

Interestingly, when one considers when most protocols in each of the above classes were designed, we find that Internet protocol designers have started with

protocols in the *Early Internet* category, and moved to ones from *Emerging Threats* when they realized that the Internet was no longer nice. However, while the core ideas did not change, the protocols were hidden behind fences such as DMZs. Since this did not suffice, they moved to *Complex Security*. As these were hard to maintain or difficult to use, we see a new trend towards *A new Simplicity*.

2.6. Systematization

In Sections 3–6 we take a closer look at each of the above classes. For each class, we identify a set of representative protocols which we analyze according to five sets of features.

Security Features: Among the essential security features that protocols should support are authentication, authorization, and use of encryption (TLS for transport layer encryption). We mark for each protocol/service which of these features is (a) in common use (●), (b) implemented but not commonly used (◦), (c) not implemented (-). If the protocol does not support a security feature we leave the space blank.

Misconfiguration traps: We consider the possible misconfiguration traps which a protocol/service may have. Under **NoAuth** we capture if (a) authentication is not offered by the protocol, (b) typically not implemented, or (c) typically not configured in the deployed service. Under **Credentials** we note if the service is commonly deployed with weak default credentials. Under **Artifacts** we note if protocol features common at design time lead to misconfigurations when used today. With **Fencing** we note those cases that depend on firewalling, etc. to ensure that they are not reachable from the Internet. We mark those protocols **NoUse** that are hardly deployed even though they replace earlier protocols, e.g., prior versions, with major misconfiguration traps.

Support: As misconfigurations often occur due to poor technical support for operators, we look closely at that for each of our representative protocols. More precisely, we look at the documentation for securely deploying the service and check if it is mostly (◦) or always (●) misleading, lacking, too complex, or otherwise insufficient. Another common aspect that leads to misconfigurations are bad defaults. But rather than looking at what can go wrong we check if the defaults are always (●) or sometimes not “sane” (◦). Sane in the sense that they enable or enforce enabling the supported security features of the protocol/service to ensure confidentiality, integrity, and availability by default.

Publications: Here we capture if problems that can lead to misconfigurations discussed under misconfiguration traps or support are already well known either in the academic world or the security community. If it is known in the academic world, we refer to a representative paper. If it is known in the community we note approximately when it became part of the system engineering lore.

Visible Instances: Next we try to estimate the number of systems, or rather IP addresses, that offer the service/protocol under discussion. We rely on different data sources, among them (a) publicly available data sets or representative papers, (b) zMap [60] scans by the authors, and (c) search results from Shodan [25], [155]. Shodan is “the world’s first search engine for Internet-connected devices”. While Shodan does provide an estimate of the possible number of IPs offering a service it is neither complete, covers all available ports [25], nor are all instances per se vulnerable.

3. The Early Internet

This class includes those protocols that were designed in the context of the early Internet, roughly from 1960 – 1988, where attacks had yet to be considered and therefore protocols were designed without security considerations. The paper by David Clark about “The Design Philosophy of the DARPA Internet Protocols” [48] does not even contain the term security even though availability in the sense of survivability is a major goal. After all, the main goal of the Internet was to interconnect existing networks with the implicit assumption that all participants worked towards the common goal of communication.

As a result attacks were not yet common. So, the need for security either did not exist or was extremely limited. Towards the end of the era, attacks against operating systems became more prominent and the first major Internet worm, namely the Morris worm, was let loose [127], [160].

Most popular protocols from that era have been updated to remain usable in today’s hostile Internet. However, this does not remove all misconfiguration traps. Today many still have artifacts of their design for a friendly Internet.

Thus, the threat model of this class is: “weak attacker” with “good enough”. The representative protocols we examine are: SMTP and DNS. Other protocols in this class include: TFTP, FTP, Finger, rexec, Chargen, NIS, RIP, NTP, WHOIS, Ident, XDMCP/X11, Syslog, rsync and IRC.

3.1. Example Protocols

FTP: The file transfer protocol (FTP) is an application layer protocol for Internet file transfer between hosts. FTP is one of the earliest Internet protocols and was first documented by RFC 114 [16] in 1971. It provides authentication within the protocol and authorization via the operating system's file system access controls. It does not offer encryption. However, it can be used over a TLS tunnel, see RFC4217 [70] from 2005.

The major misconfiguration pitfalls for FTP servers are related to either missing authentication or insufficient authorization and directory limits. These are: (a) Enable anonymous logins to share files publicly, see RFC1635 [55]. Hence, files uploaded to a server that allows anonymous access become public. Recent examples show that, e.g., private keys [41] can be exposed this way. (b) If, in addition, write access is enabled files can be deleted and/or overwritten. The FTP servers can also be abused to share malicious content. This issue has been discussed as common example by Uppuluri and Skar [169]. (c) Faulty configuration of the root-directory of an FTP server may expose system files. If, e.g., a UNIX machine exposes its global root directory, FTP users can access all files that are accessible to the user operating the FTP server. (d) FTP's dedicated data-channel enables an attacker to send files containing service commands to a remote server, e.g., SMTP. This attack can be used for amplification and firewall evasion [88]. RFC2577 [4] recommends disallowing data-channel connections to low-ports as mitigation.

To counteract some of the above threats, some modern FTP implementations, such as vsFTPD, ship a systematically locked down default configuration. It requires extensive user action to enable anonymous, write, and, non-directory-restricted access. The documentation of vsFTPD is short and precise. Other widespread FTP implementations, e.g., the versatile solution ProFTPD have a more complex documentation due to their larger feature set.

TFTP: The Trivial File Transfer Protocol (TFTP) is a "very simple protocol used to transfer files" and is documented in RFC783 [158] dated 1981. TFTP only supports reading and writing files and lacks most of the advanced features of FTP. It uses UDP as transport layer protocol. TFTP is often used for bootstrapping by providing access to files needed for system boot such as boot images, firmware updates, or network device configurations files. Revision-2, RFC1350 [157], fixed the "Sorcerer's Apprentice" protocol bug - a major data retransmission problem which leads to packet amplification.

TFTP itself does not provide authentication or authorization. It does offer limited protection by means of the operating system's file system access controls. Modifying files on an TFTP server can be restricted by most implementations. TFTP does not support encryption. TFTP is by design insecure which has also led to its most common use case, to allow bootstrapping of unprovisioned systems that therefore have no security credentials.

TFTP is also among the first protocols to suffer from unintended amplification attacks, see RFC1350 [157]. This is one of the earliest amplification attacks of stateless protocols. TFTP servers, unless shielded from general access, are subject to disclosure attacks [82]. Indeed, they often enable transitive attacks, in which an attacker first retrieves the confidential configuration files, including encryption, authentication, and authorization secrets. Then the attacker uses this information to access the systems newly configured over TFTP.

If the TFTP server allows write access, attackers can, in principle, overwrite any of the configuration files with their own, resetting passwords or configuring additional known credentials for super users. Similarly, the attacker can alter the available system images - boot as well as full system images - to include backdoors. Once provisioned they enable the attacker to take over the corresponding systems.

The documentation of the most common TFTP server implementations is short and precise and includes a sensible discussion of the security issues. Moreover, the suggested configurations are appropriate.

To counteract most of the above threats, almost all TFTP servers must be isolated from public access and subject to strict access rules or firewalls. Thus, the main misconfiguration trap is missing fences. Indeed, in 2014, MacFarlane et al. [109] found more than 600,000 publicly accessible TFTP servers. However, fencing is insufficient against inside attackers.

DNS: Since the early 1980's the Domain Name System (DNS), RFC882 and RFC883 [117], [118], is used to map hostnames to IP addresses and vice versa.

DNS is a hierarchical distributed database organized in independently administered DNS zones. These zones are implemented as subtrees in the hierarchy of the DNS. Each zone has at least one "authoritative" DNS server while one server can be authoritative for multiple zones. In addition, a nameserver can also be queried by client hosts and provide name resolution for arbitrary domains for which it is not necessarily authoritative. Most non-authoritative DNS servers only serve clients within their administrative domain, e.g., an Internet Service Provider (ISP) providing name

resolution for its customers. However, services like OpenDNS and the Google public DNS provide public available name servers.

While some services are intentionally open to the public there is a large mass of misconfigured servers unintentionally providing public name resolution. DNS by default uses connectionless UDP and usually the responses are larger than the queries as the query is contained in the response. Thus, DNS, by design, can be abused for amplification attacks, in particular both with open resolvers [148] and resolvers that return large answers.

Mitigation strategies against amplification attacks exist and are usually deployed by the large providers of open DNS servers. Nevertheless, this kind of abuse can not be completely prevented due to inherent protocol limitations [148]. While DNSSEC is currently being discussed as mitigation for various other problems in the DNS protocol suite it exacerbates this abuse as it often produces very large answers [171].

Another attack vector is information disclosure in reverse DNS lookups. Using these an attacker may infer which hosts offer which services, even inside a firewalled network, or disclose the organizational structure. Some misconfigured systems may still allow zone transfers of full DNS Zones [97], which was historically the default [14].

Currently, we find more than 10,000,000 DNS servers on the Internet. A substantial fraction, more than 5,000 [165], of these are open DNS servers of which it is unclear to what extent they deploy even the available limited amplification mitigation.

SMTP: The objective of the Simple Mail Transfer Protocol (SMTP) as documented in 1982 in RFC821 [134] and updated by RFC5231 [154] is to reliably and efficiently transfer email. Among the important features of SMTP is the ability to relay email across multiple networks.

The base architecture of SMTP used open relays for forwarding messages between Mail Transfer Agents (MTAs) without authentication or authorization, see RFC822 [52]. Authentication was suggested by an Internet draft in 1995 and added with RFC2554 [122] in 1999. TLS was also added in 1999 with RFC2487 [90].

Attackers realized early on, that open relays are great for amplifying the effects of worms, viruses, and in particular SPAM [100]. Even with the CIA features SPAM is a major daily annoyance.

These problems are usually mitigated by providing strict and well documented default configurations [96]. In today's deployments almost all SMTP servers only accept emails for their configured domains. Thus, the possibility of amplification has been reduced. If MTA

to MTA relay is allowed it is only with credentials and TLS.

Another problem with SMTP is that an attacker may take over an SMTP server and send rogue data which is not easily mitigated. The defense here is blacklisting, whitelisting, sender verification, etc. see, e.g., [49].

However, it is still possible - in an attempt to "Make Things Work" - to misconfigure SMTP servers. After all, problematic configurations do still have applicable use-cases on the Internet, e.g., an outbound email relay for a large network which every machine should use. In the wild, open SMTP relays are still observed from time to time. But most are quickly found and closed down.

NIS: NIS has been initially developed by SUN [] to have a method that ensures user identities, authentication and authorization information is present on all machines under one authoritative domain. The service is, by design, client side unauthenticated, exposes all user passwords in maximum 3DES encryption to the world and does not support transport layer encryption.

Attacks: The attacks utilizing NIS are obvious. If it is exposed on the Internet a remote attacker can dump the full user database, including the DES encrypted passwords. Reversing these is not an issue for today's hardware []. Furthermore, the attacker obtains full information on all present users.

This issue can only be addressed by appropriate firewalling — or replacing NIS with a more steady method, e.g., Kerberos. The presented issues are also not reduced by documentation indicating, that running NIS on another port is an important security feature. Still, it can be held for these howtos, that they additionally recommend iptables rules.

3.2. Discussion—Early Internet

A common misconfiguration trap in this class is the assumption that neither client authentication nor encryption is needed as the services are in a trustworthy environment. Indeed, access without authentication is considered a feature (FTP, SMTP, and DNS). Other abuses of misconfiguration are stateless amplification attacks (TFTP, DNS) where the root cause is that the server sends data without checking if the client wants it.

Based on these observations one would presume that protocols in this class have seen the end of their live cycle. However, almost all of the above protocols are still very popular. The reason is (a) the Internet relies on the services (DNS, SMTP) (b) their convenience (FTP), (c) the fact that there is no good alternative (TFTP), and (d) the service happens to be running

and is a legacy service. Worse, the implementation of, e.g., TFTP requires a small code base which makes it common in millions of customer premise equipment (CPE). Indeed, these protocols are unlikely to disappear as they are the foundation of the Internet.

The reason why such services are still in operation is twofold: Either it is presumed possible to hide the services behind firewalls (TFTP, NIS, RIP) or work on alternative protocols has started, but these protocols are not yet in Internet-wide use (DNS, SMTP). But, misconfigurations occur if either the firewall fails or the protocols are not used as originally designed.

The first major security incident which exploited a misconfiguration was an email amplification attack - namely the Morris worm in 1988 [160]. At about the same time, the community started to realize that major misconfigurations can occur, see RFC1222 [31] and RFC1223 [80] that “provide guidance for vendors, implementors, and users of Internet communication software”.

4. Emerging Threats

Security incidents such as the Morris worm changed the way that Internet protocols are perceived. Instead of designing them for the Internet at large, they became explicitly designed with firewalls in mind. Thus, network firewalls, more precisely packet filters [42], became the typical way of fencing off network services.

Bellovin and Cheswick [13] state that the motivation for Network Firewalls is: “*Computer security is a hard problem. Security on networked computers is much harder. Firewalls (barriers between two networks), when used properly, can provide a significant increase in computer security.*” In addition, the approach from 1988 onwards is, according to Bellovin and Cheswick [13]: “*Everything is guilty until proven innocent. Thus, we configure our firewalls to reject everything, unless we have explicitly made the choice - and accepted the risk - to permit it*”.

Protocol designers find network firewalls to be a convenient way to handle security. In their minds, firewalls enable them to basically ignore security threats as they presume that the firewall rejects everything that is “untrusted”. The design assumption of most protocols is that since the attacker is not strong enough to get past the firewall the protocol itself can be designed for a trusted environment.

However, as stated by Wool [175]: “*The protection that firewalls provide is only as good as the policy they are configured to implement. Analysis of real configuration data shows that corporate firewalls are*

often enforcing rule sets that violate well established security guidelines.” His conclusion is to keep firewall configurations simple but efficient to avoid misconfiguration.

Thus, the threat model for this class is: “weak attacker” with perfect security. The representative protocols we take a closer look at are: NetFlow, DHCP, and, iSCSI. Other protocols in this class include: SNMPv2, Munin, NFSv3, Wake on Lan, Remote DMA, NBD, rsyslog, SCADA, early versions of CIFS/SMB, and Mapping of Airline Traffic over Internet Protocol (MATIP).

4.1. Example Protocols

NetFlow: Cisco Systems NetFlow service allows network administrators to collect IP flow information from their network. A flow is a summary of a set of packets that pass through a device that have some common property. NetFlow uses UDP as its transport protocol. NetFlow is widely used in many ISP and enterprise networks. Indeed, many resource accounting as well as security incident systems are built on this data source. Early versions are documented as Cisco white papers. Version 9 is documented in RFC3954 [46].

NetFlow itself does not provide authentication, authorization, or encryption support. This was a conscious choice by the protocol designers, to cite RFC3954 [46]: “*The designers of NetFlow Version 9 did not impose any confidentiality, integrity or authentication requirements on the protocol because this reduced the efficiency of the implementation and it was believed at the time that the majority of deployments would confine the Flow Records to private networks, with the Collector(s) and Exporter(s) in close proximity.*” Indeed, RFC3954 specifically redirects the issue of security to the subsequent IPFIX security requirements in RFC3917 [141].

RFC3954 outlines possible attacks including disclosure of flow information data, forgery of flow records or template records, and DoS attacks on NetFlow collectors. The latter enables an attacker to exceed the collector’s storage or computational capacity and, thus, can disable the monitoring of the network. Using forgery, an attacker can inject flow information that (a) may redirect network-forensic investigations by incriminates another party or (b) lead to wrongful charges if NetFlow is the basis of accounting.

These attacks can, in principle, be mitigated by, e.g., moving to TCP and enforcing TLS/DTLS and mutual authentication. An example of such a mitigation strategy is the proposal in the IPFIX security requirements RFC3917 [141], see Section5.

The documentation of NetFlow is given mainly by Cisco White papers and Cisco device configuration examples. We observe that the documentation does not even mention how to secure NetFlow or that it is necessary. It does, however, point out that NetFlow data can be used as security enhancement to investigate network anomalies.

In summary, NetFlows main misconfiguration trap is insufficient fencing. Since NetFlow is a stateless write only protocol it is infeasible to estimate the misconfigured number of NetFlow collectors by active scans.

DHCPv4: The Dynamic Host Configuration Protocol (DHCP) is a stateful client-server protocol that can be used to provide configuration parameters to hosts - the clients - connected to the Internet. In practice, it is often used by clients to retrieve their IP address configuration as well as additional parameters including nameservers, domainnames, or local TFTP servers for diskless clients.

DHCP is based on the Bootstrap Protocol (BOOTP) and is documented in RFC1531 [57] in 1993. DHCPv6, standardized in RFC3316 [59] in 2003, tackles many of the security issues of DHCPv4. Since DHCPv4 is still commonly used for configuring IPv4 networks we discuss it in this section. In the following when we refer to DHCP we mean DHCP for IPv4 address configuration.

Regarding security RFC1531 [57] claims that “*DHCP is built directly on UDP and IP which are as yet inherently insecure*”. Since DHCP does not add any security features itself this means that the protocol lacks all basic security features. It has been designed without client authentication, server authentication, or encryption. Client Authentication was added in 2001 [58], but is not widely implemented, especially in the common embedded DHCP servers for CPEs.

As a result, any attacker can exploit the possibility for a single client to request all leases held by a DHCP server. This effectively blocks all other clients from obtaining an address. This attack is critical as it can be executed locally as well as remotely if the DHCP server is accessible from the Internet.

The next problem is that servers do not have to authenticate themselves towards the clients. Therefore, any host can pretend to be the authoritative DHCP server for its network segment. This allows an attacker to impersonate a DHCP server and send malicious information to the clients, e.g., to use (a) a different gateway which is hijacked by a monkey in the middle or (b) a different DNS server to spoof internal websites and access credentials.

Among the common DHCP servers are ISC-DHCP

and dnsmasq. Their documentation is reasonable but ignores the topic of security. The main misconfiguration trap is yet again insufficient fencing. Indeed, during the 28th Chaos Communication Congress in December 2011 the network operations team observed a DoS against their publicly reachable DHCP server. A virtual machine hosted in Amazon EC2 performed a lease starvation attack on that system [142].

Since DHCP is the common protocol for assigning dynamic IP addresses it is in common use almost everywhere. Indeed, even many home users use DHCP due to the large scale introduction of Network Address Translation (NAT) enabled home routers.

SNMPv2: The Simple Network Management Protocol (SNMP) dates back to RFC1067 [40] from 1988 and is the standard protocol for managing IP network devices, including routers, switches, workstations. It is typically shipped on the device. Small command extensions led to Community-based SNMPv2 [38] the version supported by most vendors, e.g., Cisco and Juniper. While other versions of SNMPv2 already support extensive security features most became prominent with SNMPv3 [174], [39], which we discuss in Section 5. The only authentication mechanism of SNMPv2c is the so called community string. Moreover, SNMPv2c does not support transport layer security.

Common misconfigurations are the use of weak (default) credentials, e.g., the community string `public` for read access and string `private` for the read/write access [120]. While SNMP enabled devices should be shielded by proper firewall configurations they often are not. Moreover, SNMP proxies which are designed to handle ACLs can easily be misconfigured as well. Note, weak credentials can lead to disclosure of information. With the `private` community it is possible to take over the device. Moreover, open SNMPv2 servers have been used for amplifications attacks [148].

By default, a lot of devices come with communities preconfigured, e.g., `public` or/and `private`. Even if the operator configures their own communities they often forget to remove the preconfigured ones - leaving the door open to attackers. Moreover, with a network sniffer it is possible to extract community strings. Depending on the class of the device, the documentation differs significantly from good for high-end devices to almost none for low-end customer premise devices. This is, in particular, problematic as the customer premise devices are often directly connected to the Internet. The potential number of devices that may be subject to this class of misconfiguration is, according to Shodan scans, more than 3,800,000 devices with the `public` community string.

Munin: Munin [121] is an open source networked resource monitoring tool. It is a simple service allowing retrieval of server statistics for monitoring purposes. The monitored server listens on an open port for inbound connections. The Munin monitor polls each of the targets by connecting to the port and requesting the status data. Other similar services use the same basic schema, e.g., NCSA, Ganglia, Collectd etc. Authentication is implemented by whitelisting IP addresses or address ranges of Munin servers. Authorization and transport security are not available.

Common misconfigurations are weak firewalls together with too liberal ACLs. This allows attackers to obtain detailed information about the infrastructure as well as fine-grained usage information. This, in turn, can enable a whole range of security critical side channel attacks on the cryptography of other protocols [184]. Moreover, given recent side-channel attacks that use acoustic signals from the CPU [76], it is not unlikely that attackers can use such data to extract, e.g., secret keys from the monitored servers.

While the documentation states that ACLs have to be clearly limited to authorized hosts, we still find more than 6,000 systems in the Shodan data.

NFSv3: The Network File System (NFSv3) offers transparent access to remote files. It has become one of the common UNIX network file systems. It is documented in RFC1813 [33] from 1995. NFSv4 or more general NFS with Kerberos support is discussed in Section 5.

NFSv3 offers host-based authentication on network-wide names but not per principal authentication. Moreover, NFSv3 relies on the client OS for authorization. On the wire encryption is not supported [33]. While NFSv3 in principle supports Kerberos secret keys it does not mandate them and most deployments do not use them. System lore warns to use NFS without Kerberos, see Section 6, if strong security is needed, e.g., “Kerberos is key for secure data access and not NFSv4” [167].

Common misconfigurations for NFSv3 involve the Access Control Lists: They can be too liberal, e.g., network wide, or incorrectly specified, e.g., wrong subtree of the file system. In either case, an attacker can mount an NFS share and read or modify arbitrary files. While the available documentation stresses the importance of ACLs, misconfigured servers can be found in the wild. These problems are widely known both in systems lore [62] as well as academia [166].

iSCSI: The Internet Small Computer System Interface (iSCSI) is a protocol for remotely accessing block devices over the Internet using SCSI commands first

documented 2004 in RFC3720 [150]. Since iSCSI was designed for a hostile Internet, a dedicated RFC [1] exists, that spells out the security requirements for iSCSI and similar network accessible block storage protocols. This RFC has been updated most recently in April 2014 by RFC7146 [21].

These RFCs require iSCSI to include authentication and authorization but delegate encryption and integrity to IPsec. Moreover, RFC3723 [1] acknowledges common threats for iSCSI deployments under the assumption that authentication and authorization are working, i.e., that the attacker is not able to initiate a valid connection. Moreover, the base security assumption is that there is no monkey in the middle either due to IPsec or an isolated network segment.

However, if due to a misconfiguration the iSCSI target is reachable via the Internet without authentication iSCSI becomes a severe security liability. An attacker with access to an iSCSI volume can tamper with all data thereon and can take over all machines with root file systems on those volumes.

Indeed, it is easily possible - and many large enterprise applications and howtos for setting up UNIX based targets recommend - to configure iSCSI without any authentication, neither for the whole target set nor the individual targets. Usually, this is done to cater to operational needs, especially in the cases where iSCSI volumes are used as boot devices or if a dynamic set of virtual machines has to have access to a volume. In contrast to the above common malpractice, the Payment Card Industry Data Security Standards (PCI-DSS) [131] explicitly requires client authentication, in particular, it requires it for each volume individually. Nevertheless, the common misconfiguration trap for iSCSI is missing authentication coupled with reliance on fencing.

So far, these security issues have been recognized in the industry [61] but not necessarily in academic references. With a quick zMap scan, we find roughly 9,000 iSCSI targets reachable via the Internet of which 1,000 do not require authentication. Among these are various major organizations as well as academic institutions.

4.2. Discussion—*Emerging Threats*

The base assumption of all protocols/services in this class is that the local LAN is safe. Thus, a common misconception is that they can be used with “convenient” security settings. This often leads to major security incidents when the fencing mechanisms fail. This is the major misconfiguration trap for all protocols in this class.

NetFlow is a blatant example of the low security considerations within this class. Its security concept relies entirely on fencing. DHCP goes even further: first anyone can run a rogue DHCP server and second engineers think about DHCP as a link layer protocol. SNMPv2 is one of the protocols in this class that first added security but then reduced it. NFSv3 does do authorization using OS ACLs, but without authentication. This means that anyone can impersonate anyone. While iSCSI, in principle, supports authorization and authentication some deployments do not enable it as iSCSI should be restricted to the storage network and, if used as boot device, is difficult to supply the clients with secured credentials for the iSCSI volume. However, if fencing breaks down this is a major misconfiguration as large amounts of sensible data are leaked.

However, the assumption that everything can be fenced in does not necessarily hold as specifying security policies is difficult and realizing them in a firewall is rather difficult and prone to errors [112], [2], [53], [181], [75]. Among the complications are that the designer of the security policies are not necessarily the ones that configure the firewalls and those are not necessarily the ones that deploy the network services. Moreover, updating and maintaining such rules is quite error prone. This opens up the network service for all kinds of attacks that bypass firewalls or access services that are thought to not be reachable from the Internet.

Other means of fencing include: (a) not connecting the service to the Internet at all (air gap) (b) VLANs and sub-networks (c) Virtual routing and forwarding (VRF). But, there are known attacks to all of them. Examples of how firewalls are circumvented via VPNs, hidden dialups/UMTS and other covert channels are described in the Maroochy water breach discussion [156]. Stuxnet [64] is the prime example for bypassing an air gap. One example for broken VRF is accidentally announcing a BGP full table into the VRF engine. Overall, the industrial lore states, from the Security Issues and Best Practices for Water/Wastewater Facilities [86]: *“Industrial networks are often shared with the business side of the operation. VLANs, sub-networks, firewalls all help to create a layer defense, but are not impervious.”*

This is particularly the case for services that were first envisioned for enterprises and then commonly used in Small Office/Home Office (SoHos) and home networks. In these settings security by default configurations are essential as the users often lack the knowledge and means to properly address security and network challenges.

5. Complex Security Solutions

At the end of the 20th century the awareness that firewalls were not “the” security solution became prominent. For example, RFC3365 [151] dated 2002 states: *“History has shown that applications that operate using the TCP/IP Protocol Suite wind up being used over the Internet. This is true even when the original application was not envisioned to be used in a “wide area” Internet environment. If an application isn’t designed to provide security, users of the application discover that they are vulnerable to attack.”*

As a result, protocol designers realized that (a) there was a need for improved protection architectures, e.g., the work on SANE [37] or DoS-limiting network architecture [178], and (b) that security had to be an essential feature of future protocols and services. Moreover, just adding another component to ensure security to existing protocols, e.g., firewalls, did not suffice. This fits the increased need for security in the society due to the increasing economic relevance of the Internet [110].

At the same time the diversity of the scenarios also increased with home users, SoHos, enterprises, infrastructure providers, company mergers and splits, etc. Indeed, road worriers started to appear. As a result, more assets were at stake which had to be accessible in many different ways. Thus, versatile security solutions to model complex organizational structures, e.g., via role-based access control (RBAC), were needed.

Indeed, the Danvers Doctrine [151] stated that the *“IETF should standardize on the use of the best security available”*. Thus, the threat model for this class is: “strong attacker” with “perfect security”. The representative protocols we take a closer look at are: IPP, SNMPv3, and IPFIX. Other protocols in this class include: LDAP-ACL, NFSv4, AFS, Postgresq, FTPs, RADIUS/WPA2Enterprise, s/MIME encryption, SSL/TLS, PGP, and seLinux, as an example from system security.

5.1. Example Protocols—Complex Security

IPSec: IPSec, first introduced in RFC1825–1829 [8] and updated by RFC4301–4309 [98], is a suite of protocols that promise to seamlessly extend IP with authentication, data integrity, confidentiality, non-repudiation, and protection against replay attacks.

To cite Ferguson and Schneier [66]: *“Our main criticism of IPsec is its complexity. IPsec contains too many options and too much flexibility; there are often several ways of doing the same or similar things. This is a typical committee effect.”* Thus, this is a prime

example of this class. However, IPsec is often used as an argument why it is possible to leave out certain security features in other protocols [1], [147].

IPsec is in use for corner cases such as LTE backend network security [17]. Indeed, as Ferguson and Schneier state [66]: “*Even with all the serious criticisms that we have on IPsec, it is probably the best IP security protocol available at the moment.*” Still, IPsec has not yet seen widespread deployment, e.g., [146]. One possible reason is usability. Here we cite Gutmann [78] “*If we consider security usability at all, we place it firmly in second place, and anyone wishing to dispute this claim is invited to try setting up an IPsec tunnel via a firewall or securing their email with S/MIME.*”

LDAP: LDAP, the Lightweight Directory Access Protocol, is a protocol for accessing and maintaining distributed directory information. It builds upon the ideas of X.500 but differs, in particular, with regards to security features [85] and simplicity. LDAP is designed to be extensible and flexible, see the many LDAP related RFCs, including RFC2251 [172] dated 1997, and RFC4510 [182] to RFC4519 [153]. LDAP organizes its data in a tree like ASN.1 [95]. It is often used to organize organizational information, groups, and, in particular, user data, including account information, personal information, and authentication data.

LDAP offers transport security. It also provides various forms of authentication, including SASL and Kerberos, see RFC4513 [84]. The authorization concept of LDAP is probably one of the most complex, yet, also most powerful systems currently available. In LDAP, the ACL roughly follows role-based access control [84]. It distinguishes between anonymous, authenticated, and specific connections. Specific connections are defined by properties on the object that holds the connection. This may be, but is not limited to, group membership, subtree membership, tree position, attributes in the object, etc.

The main misconfiguration opportunities for LDAP are (a) that operators use LDAP over the Internet without transport layer encryption and (b) that operators make mistakes while setting up access control [177], [68].

Especially when used for authentication, the bootstrap process is hard. Clients have to be configured and authenticated correctly, as well as authorized to access the information in the LDAP tree, and use it to perform authentication and authorization within their own applications. Furthermore, no reasonable default values can be created for ACLs, as these depend on the root nodes of the LDAP tree, which is organization dependent. To cite RFC4513 [84]: “*Operational*

experience shows that clients can (and frequently do) misuse the unauthenticated authentication mechanism of the simple Bind method see (Unauthenticated Authentication Mechanism of Simple Bind)”

Commercial distributions come with reasonable pre-configured ACLs. The non-commercial ones usually come with one rule, no write access for users that are not system administrators.

IPP: The Internet Printing Protocol (IPP) is documented in RFC2565 [89] dated 1999 and its companion RFCs. IPP is an application level protocol suite for distributed printing using Internet tools and technology. It uses HTTP, namely 1.0 or 1.1 as its transport protocol.

IPP itself implements the relevant mechanisms to perform strong authentication - by default against members of a local UNIX group via PAM (Pluggable Authentication Module) and supports the use of transport encryption. For IPP [89] authentication and authorization are critical due to an unrelated security topic, accounting. Printing, or rather, use of paper and ink, must be accounted for in most companies as well as universities. Thus, it is not surprising that the most prominent UNIX based IPP server, CUPS, currently uses TLS for all connections containing credentials. Moreover, authentication is required by default for all administrative actions.

The main misconfiguration trap with IPP is that printing on a device is - by default - allowed for unauthenticated clients [94]. Thus, a remote attacker can print on all printers they learn about. While this is usually not a major security problem, it may become an interesting basis for social engineering attacks. In addition, it enables DoS attacks, e.g., if an attacker prints endless numbers of fully black pages. Lastly, it is a nasty way of large scale resource waste. Keep in mind that the IPP service is offered by most major network attached printers by default as well as apple devices if they share their home printer.

NFSv4 with Kerberos: NFSv4 is another distributed file system protocol. Unlike its predecessor NFSv3, see Section 4, NFSv4 has support for strong security and its negotiation built in. NFSv4 uses a principal-based authentication model rather than machine-based as prior versions of NFS did.

The NFS standard [87] mandates strong security using Kerberos [185]. Kerberos, according to Neuman et al. [124], is a distributed authentication service that allows processes of a principle to prove their identity to an application server without sending data across the network that might allow an attacker to impersonate the principal. Kerberos also provides integrity and

confidentiality.

This sounds great. However, while the Storage Networking Industry Association (SNIA) states [63] that *“With careful planning, migration to NFSv4.1 and NFSv4.2 from prior versions can be accomplished without modification to applications or the supporting operational infrastructure, for a wide range of applications; home directories, HPC storage servers, backup jobs and so on.”* system lore says that NFSv4 deployment is lacking.

Indeed, NFSv4 may come with a performance penalty [44] and, as McDonald points out *“One area of great confusion is that many believe that NFSv4 requires the use of strong security. The NFSv4 specification simply states that implementation of strong RPC security by servers and clients is mandatory, not the use of strong RPC security. This misunderstanding may explain the reluctance of users to migrate to NFSv4, due to the additional work in implementing or modifying their existing Kerberos security.”*

Thus, we conclude that even though strong security options exist, often system administrators choose to not deploy them. One reason is the inherent complexity of Kerberos. Indeed, Bouillon in his Black Hat EU 2009 talk stated [30] *“However, lots of system administrators still make dramatic mistakes while configuring it (those mistakes are made more likely by buggy GUIs and their poor documentation)...”*. Indeed, from discussions with several operators we learned that they often hesitate to deploy Kerberos due to its complexity, lack of documentation, and difficulty debugging its deployment.

As a consequence it is not surprising that the Shodan scans find many NFSv3 deployments (100,000) but no NFSv4 deployments. Even though NFSv4 should be easier to detect as it uses the IANA port 2049 and does not rely on portmap.

SNMPv3: SNMPv3 [163], [39] is the successor to the Simple Network Management Protocol v2, see Section 4. The motivation for SNMPv3, RFC3410 [39] was to fix: *“The unmet goals included provision of security and administration delivering so-called “commercial grade” security with: authentication ..., privacy ...; authorization and access control; and suitable remote configuration and administration capabilities for these features.”* Thus, SNMPv3 makes few changes to the protocol aside from adding the option of on-the-wire encryption. Rather, it focuses on two main aspects, security and administration [50]. SNMPv3 supports the notion of users and authorization.

The main misconfiguration trap with SNMPv3 is using SNMPv2 instead or in parallel. Indeed, SNMPv2 and SNMPv3 are not exclusive. A host offering

Table 1. Adaption of SNMPv3 over time following [27].

protocol version	customer	devices	change from 2012 – 2013
SNMPv2	98.5%	88.2%	up 9%
SNMPv3	34.6%	10.4%	up 4%

SNMPv2 can also offer SNMPv3 even on the same port. Today many hosts indeed offer SNMPv2 and SNMPv3 simultaneously [72], [73]. Since SNMPv2 is often enabled by default the hosts remain vulnerable even though they support SNMPv3. Therefore, it is not surprising that we still see more than 3,800,000 hosts with SNMPv2 enabled. Indeed, this is supported by data from Cisco Advanced Services from May 2013 [27] which reports on the SNMP configuration status of 1,724,827 device configurations in 2013, see Table 1. Even though the adoption of SNMPv3 has increased it has been at a lower rate than SNMPv2. Part of the reason is the complexity of SNMPv3 as outlined, e.g., by Cisco training material [27].

However, even if SNMPv3 is correctly deployed, the corresponding RFCs come with another misconfiguration trap. The original RFCs, RFC2264 [23] to RFC3414 [24], only specify DES as a cipher. AES was added significantly later, RFC3826 [22]. Similarly, the only supported digest algorithms are MD5 and SHA1. This leads to implementations with weak crypto and, thus, are susceptible to advanced attacks [106]. SSHv1 suffers from similar problems as CRC32 is fixed in the protocol specification [11].

IPFix: The purpose of the IP Flow Information Export (IPFIX) protocol, see RFC5101 [47] from 2008, is to transfer IP Traffic Flow information from an exporter to a collector. IPFIX is the vendor-independent successor of NetFlow version 9, see Section 4. IPFIX supports flexible definitions of network flows via a template based extensible information model. The design goal was to make the protocol future proof as well as applicable to all network protocols.

Given the inherent insecurity of NetFlow the goal of IPFIX was to incorporate strong security in its design, see RFC3917 [141]. This resulted in RFC5101 [47] which states that IPFIX must ensure confidentiality and integrity of the transferred IPFIX data and authentication for the exporter and collector.

However, none of the major vendors, including Alcatel, Cisco, and Juniper implement any of the CIA mechanisms of the IPFIX RFC. Thus, the same exploits and misconfigurations that apply to NetFlow also apply to IPFIX, see Section 4. Hence, the major misconfiguration trap is insufficient fencing.

5.2. Discussion—Complex Security

The common design of all protocols within this class is that they are designed according to the Danvers Doctrine [151]. Therefore, they all offer full CIA support. Unfortunately, when looking at the deployment base, we either find few indications of actual use of the protocol or that the deployed instances do not utilize or implement the full CIA support.

Among the reasons are difficult setup (LDAP, NFSv4) or limited perceived benefit of latest version of the protocol (NFSv4, SNMPv3). Also, they are often used within the SoHo, rather than the enterprise, where there is a lack of required security infrastructure (IPP without Kerberos). Furthermore, implementations do not support required protocol security features (IPFIX). Furthermore, third party documentation may recommend simpler solutions, e.g., Postgres.

Another aspect is that the system administrator and the IT security professionals are typically not the same person, e.g., [29], [74]. Moreover, these systems come with substantial complexity which leads to many misconfiguration traps. For example, Xu et al. [177], [176] in their papers “*Do Not Blame Users for Misconfigurations*” and “*Hey, You Have Given Me Too Many Knobs! Understanding and Dealing with Over-Designed Configuration in System Software*” point out that major causes of today’s system failures are misconfigurations due to complexity. However, it is not necessarily the user or the system administrator who is to blame, but rather the inherent complexity and mismatch of the tools [79], [12] as well as poor usability for system operators [177], [176].

Given the above complexity, let’s review what might or might not motivate a system operator to deploy the latest secure protocol suites, see, e.g., the discussion of West as well as Scheier in their papers on “*The psychology of security*” [173], [152]: (a) No or little reward for secure behavior. Indeed, there are hardly any monetary incentives for deploying the secure versions. Some claim that such incentives might change this [77]. (b) The misconception of operators that there is a low risk of being attacked. (c) The “laziness” of the operator that wants to get the main service working first and then worry about security if there is time left. (d) The time pressure by management that forces the operator to get a service working in no time. (e) The presumption that no one is likely to get caught for not deploying the secure version of the service. (f) The maintainability of the complex security infrastructure.

6. A new Simplicity

The inherent complexity of protocol suites of the previous class provided the motivation for trade-off based security. This class is about “strong attackers” and “good enough” security. As noted by Bruce Schneier [152]: *Security is a trade-off. This is something I have written about extensively, and is a notion critical to understanding the psychology of security. There’s no such thing as absolute security, and any gain in security always involves some sort of trade-off.*

The protocol designs in this class must be seen in the context that we now have clouds as well as many start-up companies with various (mobile) applications. Indeed, the Internet is experiencing yet another growth explosion in terms of services [5], [130]. In terms of infrastructure, we are now in a “world of HTTP everywhere”, virtualization, the cloud, large scale as well as microservices architecture, and configuration orchestration.

Let us consider Internet application developers. Among the easiest ways to develop new applications is to do it in the cloud using a microservice architecture. They can rely on “Node.js” or similar programming languages and reuse existing code. The reuse of existing code has been made easy by the equivalent of appstores. As transport protocol they will most likely use HTTP/HTTPS. Moreover, the application grows as the feature sets and/or user base increases. A common observation is that the security review of such services is often lacking and the assumption is that it is possible to fence the service in a private cloud as it is just another Web service.

With “simplicity” we refer to the security concept rather than the feature sets of the protocols or, rather, protocol suites. The threat model for this class is: “strong attacker” with “good enough”. The representative protocols/protocol suites we examine are: Telnet, key-value stores, VNC, and the many incarnations of (HTTP/Socket) APIs. Other protocols in this class include: PulseAudio’s and Systemd’s internal protocols as well as control protocols for tools such as Nessus, Aircrack, etc. Since most of these use HTTP as transport layer protocols the same misconfiguration traps apply as discussed in the API subsection.

6.1. Example Protocols

Telnet: Back in 1969 the idea of the Telnet protocol, e.g., RFC15, RFC137, RFC854, RFC5198 [36], [128], [138], [101], was to make a terminal usable by a remote host as if it was local. The result was a

bi-directional, byte-oriented communication protocol. Since SSH [179] was designed, in 1995, as a secure replacement for Telnet, rlogin, etc. [180] Telnet usage should have declined to almost zero. Therefore, if at all we should have discussed Telnet in Section 3.

Unfortunately, today for many application and infrastructure systems, e.g., Customer Premise Equipment (CPEs), Storage Area Network (SAN) Devices, and what is now the Internet of Things (IoT) and Industry 4.0, Telnet is again the default choice for accessing devices [71] - with attacks skyrocketing in the past 18 months [129]. After all, Telnet is relatively simple. Thus, most of these devices already come with a built-in daemon from the original design/equipment manufacturer (ODM/OEM) that supports a subset of the Telnet protocol in their firmware templates [51]. This is the reason we discuss Telnet in this section.

The original versions of Telnet did not offer authentication/authorization or encryption. Telnet authentication defaulted to the operating system's login. Various authentication and authorization mechanisms including Kerberos and RSA were added to Telnet in 1993, see RFC1409 [28]. Adding TLS to Telnet was suggested by an Internet draft in 2000 [26]. A recent publication on vulnerabilities in telematic systems found that for all investigated devices authentication was not enabled for the Telnet interface [71].

CPEs differ from operator equipment in the sense that they are actually operated by the end users. They also differ from the typical end-user equipment in the sense that they are often preconfigured and directly reachable via the Internet. Given vendors pre-provisioning their products with known default credentials, these CPE devices can be vulnerable as soon as they are accessible over the Internet. The extent to which this can be a problem was demonstrated in 2012 by the Carna botnet [35], [104]. Another major attack used these devices to change the DNS settings of the end-users and, thus, did a monkey in the middle attack [6].

Mitigation is relatively easy. Vendors should rethink if Telnet access is necessary at all. Even if it is, it should be possible to physically turn it on/off with a small dip switch on the device - with default off. Furthermore, they should not come with no, default, or guessable credentials. Indeed, the initial credentials should be unique for each device and should not be computable from anything that is also related to the device. Various vendors already use this approach. [108]

The documentation of the Telnet service is usually very limited as the documentation usually focuses on the devices themselves and only mentions that Telnet is supported. In the past most CPE services used default

configurations with weak credentials.

Today, we find more than 10,600,000 devices with an open Telnet port according to Shodan. Indeed, the Carna botnet exploited about 1,200,000 of these devices [35]. Furthermore, a recent study by Pa et al. [129] finds that the attack volume on Telnet enabled IoT devices has increased by many orders of magnitude since 2014.

Key-Value Stores: A useful Internet service is provided by key-value stores which are widely used by companies such as Amazon, FaceBook, Digg, and Twitter [7]. Key-value stores provide a mapping between keys and their associated data and can, if implemented *In-memory*, avoid classic I/O bottlenecks. Thus, they allow quick non-relational data storage. Examples include Dynamo, MongoDB, Redis, and memcached. Initial development started around 2004. A first paper [54] reporting on “Dynamo, Amazon’s highly available key-value store” appeared in 2007. While not a protocol in themselves key-value stores are an almost universal network service and often rely on HTTP/HTTPs or JSON for their communication.

Initially, most of the key-value services did not support authentication, authorization, and/or encryption. However, over time most were augmented with support for authentication, authorization, and encryption.

To highlight the underlying assumption of the design of all of key-value stores we cite the Dynamo paper [54]: “*Other Assumptions: Dynamo is used only by Amazons internal services. Its operation environment is assumed to be non-hostile and there are no security related requirements such as authentication and authorization.*”

Most key-value stores allow for transitive attacks. When an attacker can access a central web session storage they can impersonate users and or administrators. When using key-value stores for caching, e.g., of SQL requests or of pre-compiled Just-In-Time bytecode, access to the cache can result in information disclosure or even attack code execution.

Next we give a few examples, using memcached, of the defaults when deploying key-value stores. For memcached strong authentication on the basis of SASL was introduced in 2009. Moreover, memcached now supports transport layer encryption. However, this support is lacking by default in most Unix-derivatives as the implementation of SASL was flaky in memcached and led to bugs. We surveyed the following distributions for SASL support: *Gentoo, Debian Wheezy, Ubuntu 14.04.1 LTS, Arch Linux, Centos 6, OpenBSD 5.6, FreeBSD 10*. We found that only since 2014 *Debian Wheezy* and *Ubuntu 14.04.1 LTS* started to link against SASL by default.

There are two options regarding the configuration of the listen socket, restrictive or global. Restrictive, means a specific IP, e.g., the localhost IP. 127.0.0.1, or non restrictive using 0.0.0.0. We find that for memcached only *Debian Wheezy* and *Ubuntu 14.04.1 LTS* use the restrictive default.

We find, not surprisingly that the documentation is large given the feature set of the services. Instructions for enabling the optional security features are hidden in Dynamo, misleading in memcached, and clear and simple in redis. It seems that initially the security documentation of MongoDB was also hidden. On the 10th of February 2015 more than 40.000 MongoDB databases were unprotected on the Internet. Since then a major update to the documentation took place [119].

Indeed, the weaknesses in the security models of NoSQL databases are known. While Srinivas and Nair [162] conclude that they are on a good path forward with regard to providing CIA, Okman et al. [126] point out that *“Clearly the future generations of such DBMSs need considerable development and hardening in order to provide secure environment for sensitive data which is being stored by applications (such as social networks) using them.”*

Binaryedge [18] finds that there are still more than 175K unprotected Redis/MongoDB/Memcache instances in the Internet that can be contacted from any host.

VNC and the Remote Frame Buffer Protocol: Virtual Network Computing (VNC) describes the programs and application providing the Remote Frame Buffer (RFB) Protocol, RFC6143 [145] dated 2011. RFB allows a networked computer to use a graphical user interface on a remote machine. It has two common use cases (a) remote support for end-user systems by support staff and (b) remote administration of physically inaccessible servers. The latter is, in particular, used in Linux based virtualization solutions that provide access to the VGA output of the VMs via VNC [81]. The latter is the reason why we consider VNC in this Section.

To cite from the RFB RFC6143 [145]: *“The RFB protocol as defined here provides no security beyond the optional and cryptographically weak password check described in Section 7.2.2. In particular, it provides no protection against observation of or tampering with the data stream. It has typically been used on secure physical or virtual networks.”* However, it can be enhanced with IPsec or SSH for encryption and some implementations support plain and certificate based authentication.

Since VNC is used for remote administration, it offers many opportunities for misconfiguration. An

obvious end-user system misconfiguration, weak or no passwords, opens the end-systems to easy attack. Of course, since these are usually company provided, they “should not” be unprotected in the Internet. For remote administration of SCADA systems this is worse, as access to them implies access to industrial control system [19], [164].

VNC for remote access to virtual machine consoles is often used by Linux based virtualization solutions, e.g., libvirt and Xen. These enable VNC without authentication by default. Moreover, while VNC is typically bound to localhost libvirt offers the “convenience” option of using 0.0.0.0 by uncommenting a more restrictive binding to localhost. If an attacker gets control of a VNC port they can use keyboard commands to reboot the system [145] and boot into single user mode where the credentials for the administrative account may be changed.

To mitigate this, strong, ideally key-based, authentication and built-in encryption is needed. Therefore, the protocol must be adjusted. Furthermore, the implementations should provide these security mechanisms in an easy to use manner. Shodan lists roughly 600,000 publicly available IPs running VNC. Indeed, roughly 2,000 unprotected VNC instances have been used as the basis of a “security” roulette, see the report by Stevenson [164].

APIs and Microservices: Remote procedure calls (RPCs) are a fundamental concept introduced before 1981, with the goal to make execution of code on a remote machine as simple and straightforward as on the local machine [123], [20]. Today RPCs are everywhere, but often hidden behind a different name, examples include JSON-RPC, XML-RPC, Java-RMI, SOAP, REST. Indeed, one often refers to them as application programming interfaces (APIs).

APIs are commonly used in, e.g, (a) microservices, (b) configuration interfaces, (c) mobile application services, and (d) Web applications. Microservices [125], [139], [170] reuse the traditional Unix philosophy of combining “small, sharp tools” [93]. The communication is delegated to APIs. An example of a configuration interface is the Docker API which allows operators to orchestrate applications built in docker containers via Web services [56]. Examples of mobile applications are the various APIs used in a multitude of Android and iOS apps [67], [133], [111]. Web applications often rely on client-based javascript code that calls back to APIs on the server-side [43], [34]. While APIs often use RPCs they do not have to. Some use HTTP as a transport protocol and are either RESTfull or use JSON or XML as basis. Others use

plain JSON or plain XMLRPC. Yet another group defines their own protocol often using binary encoding.

Some APIs provide some form of authentication, authorization, and encryption. However, APIs are often used in what the application designer thinks is a fenced environment. Hence, the typical use case has most security elements disabled [170], [3]. This is the reason that APIs are in this class as they basically repeat the misconceptions of *Early Internet*, namely to trust every client.

Most attacks against APIs are ones that “just” use the service. This can result in non-intended side-effects due to missing authentication and authorization, e.g., abusing the docker API, or information leakage, e.g., in mobile application APIs [67], [133], or using the back-end API rather than the client interface to overcome rate limits against brute force attacks on the original service [65]. These attacks are enabled by misconfigurations such as APIs that are Internet-wide accessible due to (a) holes in the firewall, (b) misconfigured bouncer, and (c) inside attacks via another compromised microservice [32].

We again find that, as with almost all new cool ideas, security aspects are the ones that are considered last. Thus, many APIs come with unclear documentation and/or global binds for their management APIs, e.g., docker, tomcat, JBoss [32]. Even though the meta-documentation tells reasonably well that APIs have to be secured, the practice does not adhere to this goal. Indeed, the problems are known for microservices since 2003 [139]. Moreover, good practices are known but hardly followed [125].

6.2. Discussion—A new Simplicity

In this class, we often do not have a single protocol but concepts that are realized by different protocols that all suffer from the same misconfigurations, e.g., key-value stores and APIs. Overall, we observe a clear trend towards using HTTP as application layer protocol, likely because HTTP allows middlebox traversal [140]. In theory, this opens up the possibility of “just securing HTTP” rather than having to secure a large number of Internet protocols [146]. Still, the problems of misconfiguration remain. Even if an API uses SSL for encryption, it can still be abused by an attacker [32] to disclose information. The origin of this trend is the need for complex, yet easy to deploy services.

Apart from delegating security issues to HTTP these modern services again rely on fencing. During software and system development, the main focus is on getting the service ready and not necessarily on security.

Hence, “[...] Its operation environment is assumed to be non-hostile [...]” as stated in the Amazon Dynamo paper [54].

The underlying misconception is that services can be securely fenced off as they are only used within “internal systems”. That this can lead to misconfigurations has already been shown in previous discussions. Nevertheless, many recent security incidents have led to prominent examples of data leakage (key-value stores, APIs) [18].

The motivation for delegating security to a fenced environment is the notion that security without fencing is a complex and time consuming task, impossible to achieve, and making the (backend) services unusable. Hence, the spirit during development follows the start-up culture, which includes reusing available protocols as they are (Telnet/VNC).

Fencing techniques have advanced as well, and since 2013 include the concept of containers [99]. A common misconfiguration is captured by the following idea: “If we put a service in its own container in its own dedicated VM in its own VLAN behind a firewall nothing can happen”. However, since services have to be reachable they have to allow some access. This access often turns out to be the entry point for attacks.

Containers, e.g., Docker, are a deployment and testing mechanism. The motivation is that traditional means of software distribution do not scale to the cloud as traditional testing/release procedures do not fit the rapid development cycle. This often leads to monolithic deployment of formerly modular software components. This is unmanageable if some component has to be updated, e.g., to patch a security hole. Furthermore, containers usually come preconfigured which adds additional misconfiguration traps.

Due to their experience with *Complex Security* several developers state that one should not be too strict about security and that security is “in the way” of innovation. This goes as far as e.g. Ren et al. [143] claiming that: “*Security and privacy is one fundamental obstacle to cloud computing’s success.*”.

Old mistakes are redone as the default assumption of the software developer is again “this is an esoteric scenario and not the intended use case”. APIs may release more information than intended [67], [133]. Moreover, the cloud adds the complexities of multi-party trust and the need for mutual auditability [45]. Additionally, developer guidelines with a focus on security are hard to find even if they exist.

Overall, we note that, after having had too complex security and unusable systems, we are now in a world with easily deployable solutions. However, security is often outsourced. This results in many misconfigura-

tion traps.

7. Summary

Our systematization of misconfiguration prone Internet protocols and services is based on assumptions about the attackers—weak vs. strong attacker—as well as security trade-offs—good enough vs. perfect security—in the protocols. We find that misconfiguration prone Internet protocols and their services fall into one of our four classes: *Early Internet*, *Emerging Threats*, *Complex Security*, and *A new Simplicity*.

We observe that protocol design and service development have come full circle with regard to security. In the sense that initially the Internet was a cooperative environment, see *Early Internet*. Once it was realized that the Internet was hostile it was presumed that it was possible to fence off services, see *Emerging Threats*. However, attackers are getting stronger and assets more valuable and, thus, there is the attempt to get security “right”, see *Complex Security*. The drawback is complexity and, thus, we complete the circle back to a simple security model in a presumed friendly environment guaranteed by enhanced fencing mechanisms, see *A new Simplicity*. Overall, we find many misconfiguration traps in all of the above classes.

Fencing is seen as a major security solution and often used as only barrier protecting major assets. However, fencing is a major misconfiguration trap in itself. After all, with regards to fencing there is the statement from RFC3365 [151]: “*History has shown that applications that operate using the TCP/IP Protocol Suite wind up being used over the Internet. This is true even when the original application was not envisioned to be used in a wide area Internet environment.*”

We observe that protocol/service designers are clearly forced from one extreme—full but too complex security—to the other extreme—hardly any security but easily deployable software that works “out of the box”. Indeed, our observation is that operators get frustrated if the software does not behave as it should or prevents them from doing what they want to do. Moreover, for many decision makers, security is not an end in itself but it is a cost factor that they “also have to take care of”—if it cannot be avoided. Yet, most often a security incident proves to be more costly than proactive security considerations would have been.

7.1. Lessons Learned

Besides these observations, we compile a set of action points and requirements protocol designers **MUST** consider when they create new or update old protocols.

Early Internet: The major lesson from this class is that security requirements and environments change. However, this class also demonstrates that such issues can be tackled. Hence, *when updating a protocol, one MUST purge problematic use-cases and design choices*. Only then can appeals for fixing bad configurations combined with sanctioning of insecure practices lead to improved overall security (see, e.g., [96] and [105]). While the community succeeded with SMTP, the Internet still suffers from, e.g., DNS and NTP amplification attacks. In addition, other protocols from this era still linger and urgently require revisiting.

Emerging Threats: Given that the non-hostile Internet is gone, a new paradigm emerged. Instead of adjusting the protocols to the new insecure environment, the environment is re-defined to be fenced. Thus, the assumption is that the service is behind a firewall or that lower layers offer security guarantees. Yet, it is common knowledge that firewalls tend to fail eventually and lower layers do not hold up to their promises. Thus, *when designing a protocol, one MUST not assume that it will only be used in the environment it was designed for*. Moreover, *when designing a protocol, one MUST include authentication, authorization and confidentiality preserving methods*.

Complex Security: Here the community tried to address some of the above recommendations. However, the resulting protocols come with other complications. Security features designed for an enterprise setup may not be appropriate in SoHo scenarios, e.g., recall IPP. IPSec is so complex that it is not in widespread use. Moreover, implementations do not necessarily inter-operate. In an attempt to ensure good cryptographic algorithms, SNMPv3 and SSHv1 specified algorithms that were good then but not necessarily now. Hence this class provides us with the following three lessons. First, *when designing a protocol, one MUST ensure that the security is scalable*. Scalable in the sense that it is applicable to large as well as small setups. Even the simplest setup should provide confidentiality, integrity, and availability by default. Second, *when designing a protocol, one MUST keep it simple and concise*, to allow multiple parties to implement inter-operable solutions. Third, *when designing a protocol, one MUST not mandate the use of specific ciphers but one MUST exclude plain and weak algorithms* since cryptographic algorithms become obsolete over time.

A new Simplicity: Overwhelmed by the complexity of the previous protocol generation a new protocol design trend emerged. Protocols become simple again, and designer focus on getting things done. New technologies, such as “Cloud” and “SDN” also bring new paradigms.

Table 2. Summary of all selected representative protocols.

Example		Security Features			Misconfiguration traps					Support		Publications		Visible Instances	
	Introduced	Authentication	Authorization	TLS	NoAuth	Credentials	Artifacts	Fencing	NoUse	Bad Defaults	Bad Documentation	Academia	Lore	Total	Affected
Early Internet															
FTP	1971	●	○		●	●				○	○		1999	3,000,000 ^c	
TFTP	1981							●				2015 [109]	2006		600,000 ^a [109]
SMTP	1982	○		○	●		●					2004 [96]	1995	5,600,000 ^c	
DNS	1984	-	-		●		●				○	2004 [161]	~2000	10,000,000 ^c	>5,000 ^a [165]
Emerging Threats															
NetFlow	~1990							●					2004		
DHCP	1993	-			●			●		●	●		1993		
SNMPv2	1993	○	○			●		●		●	●	2014 [148]	~2012	100,000 ^c	3,800,000 ^c
NFSv3	1995	○	○	○	●			●		○	○	1999 [83]	2015		
Munin	~2000							●							6,000 ^c
iSCSI	2002	○	○		●			●		●	○		2009		1,000 ^b
Complex Security															
IPSec	~1995	●	●	●				●		○	○	2000 [66]	2005		
LDAP	1997	○	○	●	●					○	○		2006	200,000 ^c	
IPP	1999	○	○	○	●					●	●		2003		400,000 ^c
NFSv4+Krb5	2000	●	●	●				●		●	●		2009		
SNMPv3	2002	●	●	●			●	●		○	○		2013		
IPFix	2008	-	-	-	●					○	○		2014		
A new Simplicity															
Telnet	1969	●	○		●	●		●		●	●	2014 [104]	1993	10,600,000 ^c	400,000 ^a [35]
KV Stores	2006	○	○	○				●		○	○	2011 [126]	2015		175,000 ^a [18]
APIs	2000	○	○	●	●			●		●	●	2003 [139]	2013	200,000 ^c	
VNC	1998	○		○	●	●		●		○	○	2011 [92]	2014	600,000 ^c	>2,000 ^a [19]
		●: Commonly Used ○: Uncommon -: Not Implemented : Not Specified			●: misconfiguration traps					●: Most ○: Some		Year Published [example ref.]		^a cited data ^b zMap ^c Shodan	

We find that large enterprises develop protocols very similar to those of the era of emerging threats while ignoring lessons learned often for the sake of performance. It works well for the designed environment, e.g., large enterprises where the perimeter is secure, but fails in different settings. The concept of “everything over HTTP” leads to “protocols” which ignore and/or misuse basic concepts such as authentication and in particular authorization. In addition, we find that often new actors, e.g., the automotive industry, adapt IP based technologies and re-do old mistakes. Hence, the most important takeaway from this class is that *when designing a protocol, one MUST take past*

lessons into account. While this sounds simple history tells us it is not. Indeed, almost all misconfigurations pointed out by this survey are covered in even the most basic security textbooks. Yet, they remain common and repeatedly cause major data leaks and/or outages. Thus, the prime message of this survey is to finally apply what we have learned.

General Observations: Protocol designers MUST be strict in requiring all security features from implementations and this must be reflected in the RFCs or related standards. To advance the use of TLS, a decent infrastructure, e.g., DANE [10], [91] is necessary. If such an infrastructure is unavailable encryption should

at least be performed opportunistically [180], [149].

Opportunistic encryption raises the question of usability. Usability has to become a key part of the protocol design process: The goal has to be to make the protocol secure while designing the system such that it is simple to comprehend and use. The correct way is either the only way or at least the easiest way [9], [149], [15].

Good examples of such design practices exist in the context of user centric protocols, e.g., SSH [180] and the large set of encrypted (mobile) messaging protocols [168]. The rise of the latter is motivated by the—emotional—demands of end-users [103] and professionals [115] for easy but secure communication. Important security features include strong mutual authentication and opportunistic encryption. The same kind of protocol design is needed for all Internet protocols and, in particular, those used within the infrastructure.

References

- [1] B. Aboba, J. Tseng, J. Walker, V. Rangan, and F. Travostino, “Securing Block Storage Protocols over IP,” RFC 3723 (Proposed Standard), Internet Engineering Task Force, April 2004, updated by RFC 7146. [Online]. Available: <http://www.ietf.org/rfc/rfc3723.txt>
- [2] E. S. Al-Shaer and H. H. Hamed, “Firewall policy advisor for anomaly discovery and rule editing,” in *Proc. IFIP/IEEE Symposium Integrated Network Management*, 2003, pp. 17–30.
- [3] R. Alarcón and E. Wilde, “Restler: Crawling restful services,” in *Proc. World Wide Web Conference*, 2010, pp. 1051–1052.
- [4] M. Allman and S. Ostermann, “FTP Security Considerations,” RFC 2577 (Informational), Internet Engineering Task Force, May 1999. [Online]. Available: <http://www.ietf.org/rfc/rfc2577.txt>
- [5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, “A view of cloud computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.
- [6] F. Assolini, “The Tale of One Thousand and One DSL Modems,” 2012, Kaspersky Lab.
- [7] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, “Workload analysis of a large-scale key-value store,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, 2012, pp. 53–64.
- [8] R. Atkinson, “Security Architecture for the Internet Protocol,” RFC 1825 (Proposed Standard), Internet Engineering Task Force, August 1995, obsoleted by RFC 2401. [Online]. Available: <http://www.ietf.org/rfc/rfc1825.txt>
- [9] D. Balfanz, G. Durfee, R. E. Grinter, and D. K. Smetters, “In search of usable security: Five lessons from the field,” *Proc. IEEE Security & Privacy*, no. 5, pp. 19–24, 2004.
- [10] R. Barnes, “Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE),” RFC 6394 (Informational), Internet Engineering Task Force, October 2011. [Online]. Available: <http://www.ietf.org/rfc/rfc6394.txt>
- [11] D. J. Barrett, R. E. Silverman, and R. G. Byrnes, *SSH, The Secure Shell: The Definitive Guide: The Definitive Guide*. O’Reilly Media, Inc., 2005.
- [12] R. Barrett, E. Kandogan, P. P. Maglio, E. M. Haber, L. A. Takayama, and M. Prabaker, “Field studies of computer system administrators: analysis of system management tools and practices,” in *Proc. ACM Conference on Computer Supported Cooperative Work*, 2004, pp. 388–395.
- [13] S. M. Bellovin and W. R. Cheswick, “Network firewalls,” *IEEE Communication Magazine*, vol. 32, no. 9, pp. 50–57, 1994.
- [14] D. J. Bernstein. How the AXFR protocol works. <http://cr.yp.to/djbdns/axfr-notes.html>.
- [15] D. J. Bernstein, T. Lange, and P. Schwabe, “The security impact of a new cryptographic library,” in *Progress in Cryptology—LATINCRYPT 2012*, 2012, pp. 159–176.
- [16] A. Bhushan, “File Transfer Protocol,” RFC 114, Internet Engineering Task Force, April 1971, updated by RFCs 133, 141, 171, 172. [Online]. Available: <http://www.ietf.org/rfc/rfc114.txt>
- [17] A. N. Bikos and N. Sklavos, “LTE/SAE security issues on 4G wireless networks,” *Proc. IEEE Security & Privacy*, vol. 11, no. 2, pp. 55–62, 2013.
- [18] Binary Edge. (2015, August) Data, technologies and security - part 1. <http://blog.binaryedge.io/2015/08/10/data-technologies-and-security-part-1/>.
- [19] ——. (2015, September) VNC, image analysis and data science. <http://blog.binaryedge.io/2015/09/30/vnc-image-analysis-and-data-science/>.
- [20] A. D. Birrell and B. J. Nelson, “Implementing remote procedure calls,” *ACM Trans. Computer Systems*, vol. 2, no. 1, pp. 39–59, 1984.
- [21] D. Black and P. Koning, “Securing Block Storage Protocols over IP: RFC 3723 Requirements Update for IPsec v3,” RFC 7146 (Proposed Standard), Internet Engineering Task Force, April 2014. [Online]. Available: <http://www.ietf.org/rfc/rfc7146.txt>

- [22] U. Blumenthal, F. Maino, and K. McCloghrie, “The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model,” RFC 3826 (Proposed Standard), Internet Engineering Task Force, June 2004. [Online]. Available: <http://www.ietf.org/rfc/rfc3826.txt>
- [23] U. Blumenthal and B. Wijnen, “User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3),” RFC 2264 (Proposed Standard), Internet Engineering Task Force, January 1998, obsoleted by RFC 2274. [Online]. Available: <http://www.ietf.org/rfc/rfc2264.txt>
- [24] —, “User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3),” RFC 3414 (INTERNET STANDARD), Internet Engineering Task Force, December 2002, updated by RFC 5590. [Online]. Available: <http://www.ietf.org/rfc/rfc3414.txt>
- [25] R. Bodenheimer, J. Butts, S. Dunlap, and B. Mullins, “Evaluation of the ability of the shodan search engine to identify internet-facing industrial control devices,” *Elsevier Journal of Critical Infrastructure Protection*, vol. 7, no. 2, pp. 114–123, 2014.
- [26] M. Boe and J. Altman, “TLS-based Telnet Security,” INTERNET-DRAFT draft-ietf-tn3270e-telnet-tls-06, Internet Engineering Task Force, April 2002. [Online]. Available: <https://tools.ietf.org/html/draft-ietf-tn3270e-telnet-tls-06>
- [27] G. Bollinger, “Securely managing your networks with SNMPv3,” *CiscoLIVE! BRKNMS-2658*, 2015. [Online]. Available: <https://clnv.s3.amazonaws.com/2015/usa/pdf/BRKNMS-2658.pdf>
- [28] D. Borman, “Telnet Authentication Option,” RFC 1409 (Experimental), Internet Engineering Task Force, January 1993, obsoleted by RFC 1416. [Online]. Available: <http://www.ietf.org/rfc/rfc1409.txt>
- [29] D. Botta, R. Werlinger, A. Gagné, K. Beznosov, L. Iverson, S. Fels, and B. Fisher, “Towards understanding IT security professionals and their tools,” in *Proc. ACM Symposium on Usable Privacy and Security*, 2007, pp. 100–111.
- [30] E. Bouillon, “Taming the beast: Assess kerberos-protected networks,” *Black Hat EU*, 2009.
- [31] H. Braun and Y. Rekhter, “Advancing the NSFNET routing architecture,” RFC 1222 (Informational), Internet Engineering Task Force, May 1991. [Online]. Available: <http://www.ietf.org/rfc/rfc1222.txt>
- [32] S. Breen. (2015, November) What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common? This Vulnerability. <http://foxglovesecurity.com/2015/11/06/what-do-weblogic>.
- [33] B. Callaghan, B. Pawlowski, and P. Staubach, “NFS Version 3 Protocol Specification,” RFC 1813 (Informational), Internet Engineering Task Force, June 1995. [Online]. Available: <http://www.ietf.org/rfc/rfc1813.txt>
- [34] M. Cantelon, M. Harter, T. Holowaychuk, and N. Rajlich, *Node.js in Action*. Manning, 2014.
- [35] Carna Botnet, “Internet census 2012: Port scanning/0 using insecure embedded devices,” 2013.
- [36] C. Carr, “Network subsystem for time sharing hosts,” RFC 15, Internet Engineering Task Force, September 1969. [Online]. Available: <http://www.ietf.org/rfc/rfc15.txt>
- [37] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, “SANE: A protection architecture for enterprise networks.” in *Proc. Usenix Security Symp.*, 2006.
- [38] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, “Introduction to Community-based SNMPv2,” RFC 1901 (Historic), Internet Engineering Task Force, January 1996. [Online]. Available: <http://www.ietf.org/rfc/rfc1901.txt>
- [39] J. Case, R. Mundy, D. Partain, and B. Stewart, “Introduction and Applicability Statements for Internet-Standard Management Framework,” RFC 3410 (Informational), Internet Engineering Task Force, December 2002. [Online]. Available: <http://www.ietf.org/rfc/rfc3410.txt>
- [40] J. Case, M. Fedor, M. Schoffstall, and J. Davin, “Simple Network Management Protocol,” RFC 1067, Internet Engineering Task Force, August 1988, obsoleted by RFC 1098. [Online]. Available: <http://www.ietf.org/rfc/rfc1067.txt>
- [41] A. Caudill. (2013, April) Security done wrong: Leaky FTP server. <http://adamcaudill.com/2013/04/04/security-done-wrong-leaky-ftp-server/>.
- [42] D. B. Chapman, “Network (in) security through IP packet filtering.” in *Proc. Usenix*, 1992.
- [43] A. Charland and B. Leroux, “Mobile application development: Web vs. native,” *Communications of the ACM*, vol. 54, no. 5, pp. 49–53, 2011.
- [44] M. Chen, D. Hildebrand, G. Kuenning, S. Shankaranarayana, B. Singh, and E. Zadok, “Newer is sometimes better: An evaluation of NFSv4.” *Proc. ACM SIGMETRICS*, 2015.
- [45] Y. Chen, V. Paxson, and R. H. Katz, “Whats new about cloud computing security,” *University of California, Berkeley Report No. UCB/EECS-2010-5 January*, 2010.
- [46] B. Claise, “Cisco Systems NetFlow Services Export Version 9,” RFC 3954 (Informational), Internet Engineering Task Force, October 2004. [Online]. Available: <http://www.ietf.org/rfc/rfc3954.txt>

- [47] —, “Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information,” RFC 5101 (Proposed Standard), Internet Engineering Task Force, January 2008, obsoleted by RFC 7011. [Online]. Available: <http://www.ietf.org/rfc/rfc5101.txt>
- [48] D. Clark, “The design philosophy of the DARPA Internet protocols,” *ACM Computer Communication Review*, vol. 18, no. 4, pp. 106–114, 1988.
- [49] G. V. Cormack, “Email spam filtering: A systematic review,” *Foundations and Trends in Information Retrieval*, vol. 1, no. 4, pp. 335–455, 2007.
- [50] A. Corrente and L. Tura, “Security performance analysis of SNMPv3 with respect to SNMPv2c,” in *Proc. IFIP/IEEE Network Operations and Management Symposium (NOMS)*, vol. 1, 2004, pp. 729–742.
- [51] A. Costin, J. Zaddach, A. Francillon, D. Balzarotti, and S. Antipolis, “A large-scale analysis of the security of embedded firmwares,” in *Proc. Usenix Security Symp.*, 2014.
- [52] D. Crocker, “STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES,” RFC 822 (INTERNET STANDARD), Internet Engineering Task Force, August 1982, obsoleted by RFC 2822, updated by RFCs 1123, 2156, 1327, 1138, 1148. [Online]. Available: <http://www.ietf.org/rfc/rfc822.txt>
- [53] F. Cuppens, N. Cuppens-Boullahia, and J. Garcia-Alfaro, “Detection and removal of firewall misconfiguration,” in *Proc. IASTED Conference on Communication, Network and Information Security*, vol. 1, 2005, pp. 154–162.
- [54] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Voshall, and W. Vogels, “Dynamo: Amazon’s highly available key-value store,” in *ACM SIGOPS Operating System Review*, vol. 41, no. 6, 2007, pp. 205–220.
- [55] P. Deutsch, A. Emtage, and A. Marine, “How to Use Anonymous FTP,” RFC 1635 (Informational), Internet Engineering Task Force, May 1994. [Online]. Available: <http://www.ietf.org/rfc/rfc1635.txt>
- [56] Docker.com. (2015) <http://docker.io>.
- [57] R. Droms, “Dynamic Host Configuration Protocol,” RFC 1531 (Proposed Standard), Internet Engineering Task Force, October 1993, obsoleted by RFC 1541. [Online]. Available: <http://www.ietf.org/rfc/rfc1531.txt>
- [58] R. Droms and W. Arbaugh, “Authentication for DHCP Messages,” RFC 3118 (Proposed Standard), Internet Engineering Task Force, June 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3118.txt>
- [59] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, “Dynamic Host Configuration Protocol for IPv6 (DHCPv6),” RFC 3315 (Proposed Standard), Internet Engineering Task Force, July 2003, updated by RFCs 4361, 5494, 6221, 6422, 6644, 7083, 7227, 7283. [Online]. Available: <http://www.ietf.org/rfc/rfc3315.txt>
- [60] Z. Durumeric, E. Wustrow, and J. A. Halderman, “Zmap: Fast internet-wide scanning and its security applications,” in *Proc. Usenix Security Symp.*, 2013, pp. 605–620.
- [61] H. Dwivedi, “iSCSI security,” *Black Hat*, 2005.
- [62] M. Eisler, “NFS Version 2 and Version 3 Security Issues and the NFS Protocol’s Use of RPCSEC_GSS and Kerberos V5,” RFC 2623 (Proposed Standard), Internet Engineering Task Force, June 1999. [Online]. Available: <http://www.ietf.org/rfc/rfc2623.txt>
- [63] Ethernet Storage Forum, “An updated overview of NFSv4,” SNIA — Storage Networking Industry Association, Tech. Rep., 2015.
- [64] N. Falliere, L. O. Murchu, and E. Chien, “W32.stuxnet dossier,” *White paper, Symantec Corp., Security Response*, vol. 5, 2011.
- [65] R. Fallon, “Celebgate: Two methodological approaches to the 2014 celebrity photo hacks,” in *Internet Science*, 2015, pp. 49–60.
- [66] N. Ferguson and B. Schneier, “A cryptographic evaluation of IPsec,” 2000.
- [67] T. Fiebig, W. Katz, and J. van Beek, “Grindr application security evaluation report,” 2013. [Online]. Available: https://www.os3.nl/_media/reports/grindr.pdf
- [68] A. Findlay, “Best practices in LDAP security,” 2011.
- [69] H. Flanagan and S. Ginoza, “RFC Style Guide,” RFC 7322 (Informational), Internet Engineering Task Force, September 2014. [Online]. Available: <http://www.ietf.org/rfc/rfc7322.txt>
- [70] P. Ford-Hutchinson, “Securing FTP with TLS,” RFC 4217 (Proposed Standard), Internet Engineering Task Force, October 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4217.txt>
- [71] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, “Fast and vulnerable: A story of telematic failures,” in *Proc. USENIX Workshop on Offensive Technologies (WOOT)*, 2015.
- [72] R. Frye, D. Levi, S. Routhier, and B. Wijnen, “Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework,” RFC 2576 (Proposed Standard), Internet Engineering Task Force, March 2000, obsoleted by RFC 3584. [Online]. Available: <http://www.ietf.org/rfc/rfc2576.txt>

- [73] —, “Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework,” RFC 3584 (Best Current Practice), Internet Engineering Task Force, August 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3584.txt>
- [74] S. M. Furnell, N. Clarke, R. Werlinger, K. Hawkey, and K. Beznosov, “An integrated view of human, organizational, and technological challenges of it security management,” *Information Management & Computer Security*, vol. 17, no. 1, pp. 4–19, 2009.
- [75] J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, S. Martinez, and J. Cabot, “Management of stateful firewall misconfiguration,” *Elsevier Computers & Security*, vol. 39, pp. 64–85, 2013.
- [76] D. Genkin, A. Shamir, and E. Tromer, “RSA key extraction via low-bandwidth acoustic cryptanalysis,” in *Proc. Advances in Cryptology (CRYPTO)*, 2014, pp. 444–461.
- [77] S. J. Greenwald, K. G. Olthoff, V. Raskin, and W. Ruch, “The user non-acceptance paradigm: INFOSEC’s dirty little secret,” in *Proc. ACM Workshop on New Security Paradigms*, 2004, pp. 35–43.
- [78] P. Gutmann and I. Grigg, “Security usability,” *Proc. IEEE Security & Privacy*, vol. 3, no. 4, pp. 56–58, 2005.
- [79] E. M. Haber and J. Bailey, “Design guidelines for system administration tools developed through ethnographic field studies,” in *Proc. ACM Symposium on Computer Human Interaction for the Management of Information Technology*, 2007, p. 1.
- [80] J. Halpern, “OSI CLNS and LLC1 protocols on Network Systems HYPERchannel,” RFC 1223 (Informational), Internet Engineering Task Force, May 1991. [Online]. Available: <http://www.ietf.org/rfc/rfc1223.txt>
- [81] M. J. Hammel, “Managing KVM deployments with virt-manager,” *Linux Journal*, vol. 2011, no. 201, 2011.
- [82] M. Handley, E. Rescorla, and IAB, “Internet Denial-of-Service Considerations,” RFC 4732 (Informational), Internet Engineering Task Force, December 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4732.txt>
- [83] B. Harris and R. Hunt, “TCP/IP security threats and attack methods,” *Elsevier Computer Communications*, vol. 22, no. 10, pp. 885–897, 1999.
- [84] R. Harrison, “Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms,” RFC 4513 (Proposed Standard), Internet Engineering Task Force, June 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4513.txt>
- [85] V. Hassler, “X.500 and LDAP security: A comparative overview,” *IEEE Network Magazine*, vol. 13, no. 6, pp. 54–64, 1999.
- [86] J. Hayes, “Security issues and best practices for water/wastewater facilities,” *Proceedings of the Water Environment Federation*, vol. 2013, no. 8, pp. 6442–6461, 2013.
- [87] T. Haynes and D. Noveck, “Network File System (NFS) Version 4 Protocol,” RFC 7530 (Proposed Standard), Internet Engineering Task Force, March 2015. [Online]. Available: <http://www.ietf.org/rfc/rfc7530.txt>
- [88] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, and R. Lutz, “A software fault tree approach to requirements analysis of an intrusion detection system,” *Springer Requirements Engineering*, vol. 7, no. 4, pp. 207–220, 2002.
- [89] R. Herriot, S. Butler, P. Moore, and R. Turner, “Internet Printing Protocol/1.0: Encoding and Transport,” RFC 2565 (Experimental), Internet Engineering Task Force, April 1999, obsoleted by RFC 2910. [Online]. Available: <http://www.ietf.org/rfc/rfc2565.txt>
- [90] P. Hoffman, “SMTP Service Extension for Secure SMTP over TLS,” RFC 2487 (Proposed Standard), Internet Engineering Task Force, January 1999, obsoleted by RFC 3207. [Online]. Available: <http://www.ietf.org/rfc/rfc2487.txt>
- [91] P. Hoffman and J. Schlyter, “The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA,” RFC 6698 (Proposed Standard), Internet Engineering Task Force, August 2012, updated by RFC 7218. [Online]. Available: <http://www.ietf.org/rfc/rfc6698.txt>
- [92] H. Holm, T. Sommestad, J. Almroth, and M. Persson, “A quantitative evaluation of vulnerability scanning,” *Information Management & Computer Security*, vol. 19, no. 4, pp. 231–247, 2011.
- [93] A. Hunt and D. Thomas, *The pragmatic programmer: From journeyman to master*. Addison-Wesley Professional, 2000.
- [94] S. Institute. (2003) Printer insecurity: Is it really an issue? <https://www.sans.org/reading-room/whitepapers/threats/printer-insecurity-issue-1149>.
- [95] “Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation,” ITU-T Recommendation X.680 (07/02), ISO/IEC 8824-1:2002, 2002.
- [96] J. Jung and E. Sit, “An empirical study of spam traffic and the use of DNS black lists,” in *Proc. ACM Internet Measurement Conference*. ACM, 2004, pp. 370–375.
- [97] A. J. Kalafut, C. A. Shue, and M. Gupta, “Understanding implications of DNS zone provisioning,” in *Proc. ACM Internet Measurement Conference*, 2008, pp. 211–216.

- [98] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301 (Proposed Standard), Internet Engineering Task Force, December 2005, updated by RFC 6040. [Online]. Available: <http://www.ietf.org/rfc/rfc4301.txt>
- [99] T. Kim and N. Zeldovich, "Practical and effective sandboxing for non-root users." in *Proc. Usenix*, 2013, pp. 139–144.
- [100] J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker, "SMTP Service Extensions," RFC 1869 (INTERNET STANDARD), Internet Engineering Task Force, November 1995, obsoleted by RFC 2821. [Online]. Available: <http://www.ietf.org/rfc/rfc1869.txt>
- [101] J. Klensin and M. Padlipsky, "Unicode Format for Network Interchange," RFC 5198 (Proposed Standard), Internet Engineering Task Force, March 2008. [Online]. Available: <http://www.ietf.org/rfc/rfc5198.txt>
- [102] J. I. Krämer, "Why cryptography should not rely on physical attack complexity," Ph.D. dissertation, Springer, 2015.
- [103] L. Kraus, T. Fiebig, V. Miruchna, S. Möller, and A. Shabtai, "Analyzing end-users knowledge and feelings surrounding smartphone security and privacy," *Proc. IEEE Security & Privacy Workshops - Mobile Security Technologies (MoST)*, 2015.
- [104] T. Krenc, O. Hohlfeld, and A. Feldmann, "An Internet census taken by an illegal botnet: A qualitative assessment of published measurements," *ACM Computer Communication Review*, vol. 44, no. 3, pp. 103–111, 2014.
- [105] M. Kühner, T. Hupperich, C. Rossow, and T. Holz, "Exit from hell? Reducing the impact of amplification DDoS attacks," in *Proc. Usenix Security Symp.*, 2014.
- [106] N. Lawrence and P. Traynor, "Under new management: Practical attacks on SNMPv3," in *Proc. USENIX Workshop on Offensive Technologies (WOOT)*, 2012.
- [107] Y. Liu, A. Sarabi, J. Zhang, P. NAGHIZADEH ARD-ABILI, M. Karir, M. Bailey, and M. Liu, "Cloudy with a chance of breach: Forecasting cyber security incidents," in *Proc. Usenix Security Symp.*, 2015.
- [108] E. N. Lorente, C. Meijer, and R. Verdult, "Scrutinizing WPA2 password generating algorithms in wireless routers," in *Proc. USENIX Workshop on Offensive Technologies (WOOT)*, 2015.
- [109] R. Macfarlane and W. J. Buchanan, "Evaluation of TFTP DDoS amplification attack," *Elsevier Computers & Security*, 2015.
- [110] B. Mahadevan, "Business models for Internet-based e-commerce," *California management review*, vol. 42, no. 4, pp. 55–69, 2000.
- [111] M. Masse, *REST API design rulebook*. O'Reilly Media, Inc., 2011.
- [112] A. Mayer, A. Wool, and E. Ziskind, "Fang: A firewall analysis engine," in *Proc. IEEE Security & Privacy*, 2000, pp. 177–187.
- [113] G. McGraw, "Software security," *Proc. IEEE Security & Privacy*, vol. 2, no. 2, pp. 80–83, 2004.
- [114] —, *Software security: building security in*. Addison-Wesley Professional, 2006, vol. 1.
- [115] S. E. McGregor, P. Charters, T. Holliday, and F. Roesner, "Investigating the computer security practices and needs of journalists," in *Proc. Usenix Security Symp.*, 2015, pp. 399–414.
- [116] B. Meixell and E. Forner, "Out of control: Demonstrating scada exploitation," *Black Hat*, 2013.
- [117] P. Mockapetris, "Domain names: Concepts and facilities," RFC 882, Internet Engineering Task Force, November 1983, obsoleted by RFCs 1034, 1035, updated by RFC 973. [Online]. Available: <http://www.ietf.org/rfc/rfc882.txt>
- [118] —, "Domain names: Implementation specification," RFC 883, Internet Engineering Task Force, November 1983, obsoleted by RFCs 1034, 1035, updated by RFC 973. [Online]. Available: <http://www.ietf.org/rfc/rfc883.txt>
- [119] MongoDB website. (2015, February) <https://www.mongodb.com/blog/post/mongodb-security-best-practices>.
- [120] R. Moonen and C. D. I. BV, "Digitale achterdeuren in de nederlandse internet infrastructuur," *Itsx bv*, 2012.
- [121] Munin. (2002) An open source networked resource monitoring tool. munin-monitoring.org.
- [122] J. Myers, "SMTP Service Extension for Authentication," RFC 2554 (Proposed Standard), Internet Engineering Task Force, March 1999, obsoleted by RFC 4954. [Online]. Available: <http://www.ietf.org/rfc/rfc2554.txt>
- [123] B. J. Nelson, "Remote procedure call," 1981.
- [124] B. C. Neuman and T. Ts' O, "Kerberos: An authentication service for computer networks," *IEEE Computer Magazine*, vol. 32, no. 9, pp. 33–38, 1994.
- [125] S. Newman, *Building Microservices*. O'Reilly Media, Inc., 2015.
- [126] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, and J. Abramov, "Security issues in NoSQL databases," in *Proc. IEEE Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2011, pp. 541–547.

- [127] H. Orman, "The morris worm: A fifteen-year perspective," *Proc. IEEE Security & Privacy*, no. 5, pp. 35–43, 2003.
- [128] T. O'Sullivan, "Telnet Protocol - a proposed document," RFC 137, Internet Engineering Task Force, April 1971, updated by RFC 139. [Online]. Available: <http://www.ietf.org/rfc/rfc137.txt>
- [129] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: Analysing the rise of iot compromises," in *Proc. USENIX Workshop on Offensive Technologies (WOOT)*, 2015.
- [130] G. Pallis, "Cloud computing: the new frontier of Internet computing," *IEEE Internet Computing*, no. 5, pp. 70–73, 2010.
- [131] Payment Card Industry, "Security standards council," *Payment Card Industry Data Security Standard: Requirements and Security Assessment Procedures*, 2014.
- [132] C. P. Pfleeger and S. L. Pfleeger, *Security in computing*. Prentice Hall Professional Technical Reference, 2002.
- [133] I. Polakis, G. Argyros, T. Petsios, S. Sivakorn, and A. D. Keromytis, "Where's wally?: Precise user discovery attacks in location proximity services," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2015, pp. 817–828.
- [134] J. Postel, "Simple Mail Transfer Protocol," RFC 821 (INTERNET STANDARD), Internet Engineering Task Force, August 1982, obsoleted by RFC 2821. [Online]. Available: <http://www.ietf.org/rfc/rfc821.txt>
- [135] —, "Introduction to the STD Notes," RFC 1311 (Informational), Internet Engineering Task Force, March 1992. [Online]. Available: <http://www.ietf.org/rfc/rfc1311.txt>
- [136] —, "Instructions to RFC Authors," RFC 1543 (Informational), Internet Engineering Task Force, October 1993, obsoleted by RFC 2223. [Online]. Available: <http://www.ietf.org/rfc/rfc1543.txt>
- [137] J. Postel and J. Reynolds, "Instructions to RFC Authors," RFC 2223 (Informational), Internet Engineering Task Force, October 1997, obsoleted by RFC 7322, updated by RFCs 5741, 6949. [Online]. Available: <http://www.ietf.org/rfc/rfc2223.txt>
- [138] —, "Telnet Protocol Specification," RFC 854 (INTERNET STANDARD), Internet Engineering Task Force, May 1983, updated by RFC 5198. [Online]. Available: <http://www.ietf.org/rfc/rfc854.txt>
- [139] I. M. P. Pratistha, N. Nicoloudis, and S. Cuce, "A micro-services framework on mobile devices." in *ICWS*, 2003, pp. 320–325.
- [140] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *ACM Computer Communication Review*, vol. 43, no. 4, 2013, pp. 27–38.
- [141] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," RFC 3917 (Informational), Internet Engineering Task Force, October 2004. [Online]. Available: <http://www.ietf.org/rfc/rfc3917.txt>
- [142] K. Rechthien and W. Hargrave. (2011, December) 28C3 NOC Review. ftp://media.ccc.de/congress/2011/mp4-h264-HQ/28c3-4927-en-noc_review_28c3_camp_h264.mp4.
- [143] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, no. 1, pp. 69–73, 2012.
- [144] E. Rescorla and B. Korver, "Guidelines for Writing RFC Text on Security Considerations," RFC 3552 (Best Current Practice), Internet Engineering Task Force, July 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3552.txt>
- [145] T. Richardson and J. Levine, "The Remote Framebuffer Protocol," RFC 6143 (Informational), Internet Engineering Task Force, March 2011. [Online]. Available: <http://www.ietf.org/rfc/rfc6143.txt>
- [146] P. Richter, N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger, "Distilling the internet's application mix from packet-sampled traffic," in *Proc. Passive and Active Measurement (PAM)*, 2015, pp. 179–192.
- [147] A. Romanow, J. Mogul, T. Talpey, and S. Bailey, "Remote Direct Memory Access (RDMA) over IP Problem Statement," RFC 4297 (Informational), Internet Engineering Task Force, December 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4297.txt>
- [148] C. Rossow, "Amplification hell: Revisiting network protocols for DDoS abuse," in *Symposium on Network and Distributed System Security (NDSS)*, 2014.
- [149] V. Roth, T. Straub, and K. Richter, "Security and usability engineering with particular attention to electronic mail," *International Journal of Human-Computer Studies*, vol. 63, no. 1, pp. 51–73, 2005.
- [150] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)," RFC 3720 (Proposed Standard), Internet Engineering Task Force, April 2004, obsoleted by RFC 7143, updated by RFCs 3980, 4850, 5048, 7146. [Online]. Available: <http://www.ietf.org/rfc/rfc3720.txt>
- [151] J. Schiller, "Strong Security Requirements for Internet Engineering Task Force Standard Protocols," RFC 3365 (Best Current Practice), Internet Engineering Task Force, August 2002. [Online]. Available: <http://www.ietf.org/rfc/rfc3365.txt>

- [152] B. Schneier, "The psychology of security," in *Progress in Cryptology—AFRICACRYPT 2008*. Springer, 2008, pp. 50–79.
- [153] A. Sciberras, "Lightweight Directory Access Protocol (LDAP): Schema for User Applications," RFC 4519 (Proposed Standard), Internet Engineering Task Force, June 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4519.txt>
- [154] W. Segmuller and B. Leiba, "Sieve Email Filtering: Relational Extension," RFC 5231 (Proposed Standard), Internet Engineering Task Force, January 2008. [Online]. Available: <http://www.ietf.org/rfc/rfc5231.txt>
- [155] Shodan. (2015, November) The search engine for the Internet of Things. <https://www.shodan.io/>.
- [156] J. Slay and M. Miller, *Lessons learned from the maroochy water breach*. Springer, 2008.
- [157] K. Sollins, "The TFTP Protocol (Revision 2)," RFC 1350 (INTERNET STANDARD), Internet Engineering Task Force, July 1992, updated by RFCs 1782, 1783, 1784, 1785, 2347, 2348, 2349. [Online]. Available: <http://www.ietf.org/rfc/rfc1350.txt>
- [158] —, "TFTP Protocol (revision 2)," RFC 783, Internet Engineering Task Force, June 1981, obsoleted by RFC 1350. [Online]. Available: <http://www.ietf.org/rfc/rfc783.txt>
- [159] O. Solon. (2014, March) Nhs patient data made publicly available online. <http://www.wired.co.uk/news/archive/2014-03/03/care-data-leaks>.
- [160] E. H. Spafford, "The Internet worm program: An analysis," *ACM Computer Communication Review*, vol. 19, no. 1, pp. 17–57, 1989.
- [161] S. M. Specht and R. B. Lee, "Distributed denial of service: Taxonomies of attacks, tools, and countermeasures." in *ISCA PDCS*, 2004, pp. 543–550.
- [162] S. Srinivas and A. Nair, "Security maturity in NoSQL databases—are they secure enough to haul the modern it applications?" in *Proc. IEEE Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015, pp. 739–744.
- [163] W. Stallings, "SNMPv3: A security enhancement for SNMP," *IEEE Communications Surveys*, vol. 1, no. 1, pp. 2–17, 1998.
- [164] K. Stevenson. (2014, December) Open season on vnc servers around the world. <https://medium.com/@kylestev/open-season-on-vnc-servers-around-the-world-4b89a0f8d992>.
- [165] F. Streibelt, J. Böttger, N. Chatzis, G. Smaragdakis, and A. Feldmann, "Exploring EDNS-client-subnet adopters in your free time," in *Proc. ACM Internet Measurement Conference*, 2013, pp. 305–312.
- [166] A. S. Tanenbaum and M. Van Steen, *Distributed systems*. Prentice Hall, 2007.
- [167] U. Troppens. (2014) Secure data access with kerberized nfs. https://www.ibm.com/developerworks/community/blogs/storageneers/entry/secure_data_access_with_kerberized_nfs?lang=en.
- [168] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "Sok: Secure messaging," in *Proc. IEEE Security & Privacy*, 2015, pp. 232–249.
- [169] P. Uppuluri and R. Sekar, "Experiences with specification-based intrusion detection," in *Proc. Recent Advances in Intrusion Detection*, 2001, pp. 172–189.
- [170] A. Van Halteren and P. Pawar, "Mobile service platform: A middleware for nomadic mobile service provisioning," in *Proc. IEEE Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2006, pp. 292–299.
- [171] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "DNSSEC and its potential for DDoS attacks: A comprehensive measurement study," in *Proc. ACM Internet Measurement Conference*, 2014.
- [172] M. Wahl, T. Howes, and S. Kille, "Lightweight Directory Access Protocol (v3)," RFC 2251 (Proposed Standard), Internet Engineering Task Force, December 1997, obsoleted by RFCs 4510, 4511, 4513, 4512, updated by RFCs 3377, 3771. [Online]. Available: <http://www.ietf.org/rfc/rfc2251.txt>
- [173] R. West, "The psychology of security," *Communications of the ACM*, vol. 51, no. 4, pp. 34–40, 2008.
- [174] B. Wijnen, D. Harrington, and R. Presuhn, "An Architecture for Describing SNMP Management Frameworks," RFC 2571 (Draft Standard), Internet Engineering Task Force, April 1999, obsoleted by RFC 3411. [Online]. Available: <http://www.ietf.org/rfc/rfc2571.txt>
- [175] A. Wool, "A quantitative study of firewall configuration errors," *IEEE Computer*, vol. 37, no. 6, pp. 62–67, 2004.
- [176] T. Xu, L. Jin, X. Fan, Y. Zhou, S. Pasupathy, and R. Talwadker, "Hey, you have given me too many knobs!: Understanding and dealing with over-designed configuration in system software," in *Proc. ACM Meeting on Foundations of Software Engineering*, 2015, pp. 307–319.
- [177] T. Xu, J. Zhang, P. Huang, J. Zheng, T. Sheng, D. Yuan, Y. Zhou, and S. Pasupathy, "Do not blame users for misconfigurations," in *Proc. ACM Conference on Symposium on Operating Systems Principles (SOSP)*, 2013, pp. 244–259.
- [178] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," in *ACM Computer Communication Review*, vol. 35, no. 4, 2005, pp. 241–252.

- [179] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture," RFC 4251 (Proposed Standard), Internet Engineering Task Force, January 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4251.txt>
- [180] T. Ylönen, "SSH: Secure login connections over the Internet," in *Proc. Usenix Security Symp.*, 1996.
- [181] L. Yuan, H. Chen, J. Mai, C.-N. Chuah, Z. Su, and P. Mohapatra, "Fireman: A toolkit for firewall modeling and analysis," in *Proc. IEEE Security & Privacy*, 2006, pp. 15–pp.
- [182] K. Zeilenga, "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map," RFC 4510 (Proposed Standard), Internet Engineering Task Force, June 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4510.txt>
- [183] J. Zhang, Z. Durumeric, M. Bailey, M. Liu, and M. Karir, "On the mismanagement and maliciousness of networks," in *Proc. Symposium on Network and Distributed System Security (NDSS)*, 2014.
- [184] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM side channels and their use to extract private keys," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2012, pp. 305–316.
- [185] L. Zhu, K. Jaganathan, and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2," RFC 4121 (Proposed Standard), Internet Engineering Task Force, July 2005, updated by RFCs 6112, 6542, 6649. [Online]. Available: <http://www.ietf.org/rfc/rfc4121.txt>