CyBOK

NETWORK SECURITY KNOWLEDGE AREA

Issue 1.0

AUTHOR: Sanjay Jha – University of New South Wales

EDITOR: Andrew Martin – Oxford University

REVIEWERS:

Gene Tsudik – University of California

Shishir Nagaraja – University of Strathclyde

*Additional review pending

Network Security

Sanjay Jha

January 2019

INTRODUCTION

The ubiquity of the Internet allows us to connect all sorts of devices to the network and gain unprecedented access to a whole range of applications and services anytime, anywhere. However, our heavy reliance on networking technology also makes it an attractive target for malicious users who are willing to compromise the security of our communications and/or cause disruption to services that are critical for our day-to-day survival in a connected world. In this chapter, we will explain the challenges associated with securing a network under a variety of attacks for a number of networking technologies and widely used security protocols, along with emerging security challenges and solutions. This chapter aims to provide the necessary background in order to understand other knowledge areas, in particular the Security Operations and Incidence Management KA SOIM which takes a more holistic view of security and deals with operational aspects. An understanding of basic networking protocol stack and TCP/IP suite is assumed. Basic networking text books explain the fundamentals of the 7-layer ISO OSI model and Internet Protocol [1]. When considering the security of the Internet and wireless LAN (WLAN) technologies, it can sometimes be instructive to consider how certain original protocols are either designed without bearing security in mind, or with poor security design decisions. This is not merely of historical interest: contemporary designs are often constrained by their predecessors for pragmatic reasons.

CONTENT

1 Internet Architecture

A complex system such as distributed applications running over a range of networking technologies is best understood when viewed as layered architecture. Figure 1 shows the 7-layer protocol ISO OSI stack and the interaction between the various layers. The model also allows us to understand the security issues on each layer and the interplay between them. The Internet is the predominant architecture today. However, it uses only five layers from the protocol stack in figure 1 i.e., layers 1-4 and layer 7. The Presentation and Session layers shown in the dotted box are optional in the IP protocol stack and some or all of the functions can be custom built on application requirements. Network security requires cryptographic techniques such as public and symmetric keys for encryption and signing, block and stream ciphers, hashing, and digital signature, as described in the Cryptography Knowledge Area. We will take an applied approach to understand how these techniques help build a secure network.

2 Network Protocols and Vulnerability

Typically, the Dolev-Yao, [2] adversarial formal model is used for a formal analysis of security protocols in the research literature. The Dolev-Yao model assumes that an adversary has complete control over the entire network, and concurrent executions of the protocol between the same set of 2-or-more parties can take place. The Dolev-Yao model describes the worst possible adversary: depending on

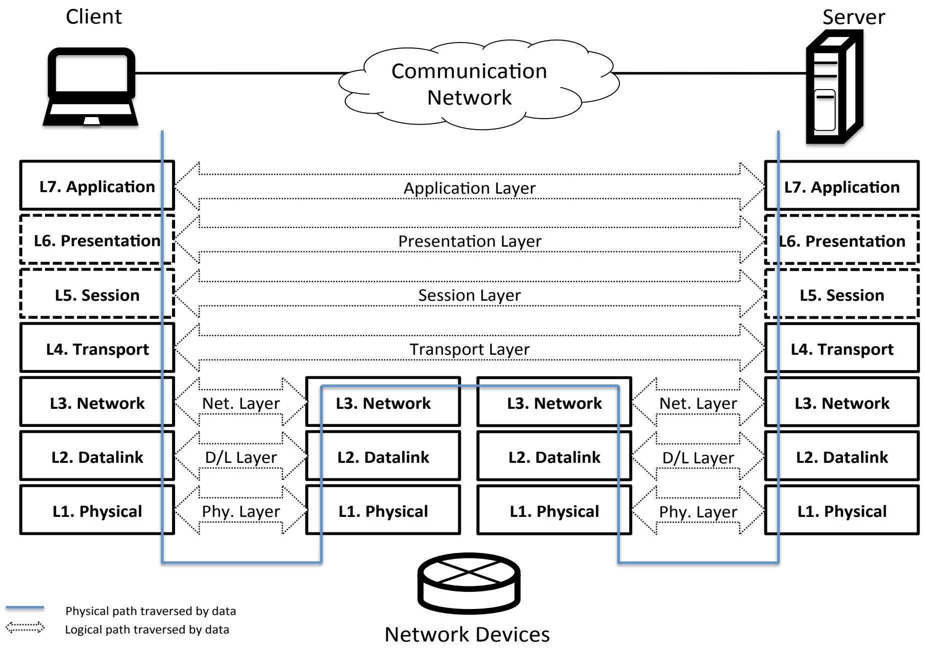

Figure 1: 7 Layer Protocol Stack

the context, real adversaries may have limited capabilities. This model is summarised as allowing the adversary to read any message, prevent delivery of any message, duplicate any message, or otherwise synthesise any message for which the adversary has the relevant cryptographic keys (if any).

We examine a few common network security attacks to highlight the importance of understanding network security issues. The popular characters called Alice and Bob from the security literature want to exchange messages securely. In terms of information and communication infrastructure context, we can replace Alice and Bob with Web servers and clients, two email clients, two people using video-conferencing and so on. The hackers, an eavesdropper called Eve, and a malicious attacker called Mallory are waiting to compromise their communications. Messages sent by Alice and Bob over a network can be captured by Eve using packet sniffing tools. This allows Eve to inspect each packet and possibly extract confidential information such as passwords, credit card details and many other types of sensitive information. Broadcast networking technologies such as wireless LAN or cable modem make it relatively easy to sniff packets. The man in the middle attack (MiTM) is another common security threat where Mallory, an attacker, places himself between Alice and Bob. For example, a compromised gateway/router/access-point, malware present in the user's device or server can potentially capture all of the packets being exchanged between the two parties, add/modify/delete information and carry out other malicious activities. The Denial of Service (DoS) attack is a technique where an attacker sends an avalanche of bogus packets to a server. This would either keep the server constantly busy or clog up the access link, resulting in disruption of service for legitimate users. Typically, a large number of compromised hosts (bots) are used to launch a distributed DoS attack, aka DDoS. DoS and MiTM are not disjoint; many DoS attacks are also MiTM, and vice-versa. Mirai [3] is an example of a malware, first found in 2016, which launched a DDoS attack by compromising Linux-based consumer devices, aka Internet of Things (IoT) devices, such as IP cameras, utility meters, home routers and others. The IoT devices were turned into bots by exploiting weak authentication configurations including use of default passwords. The bots were then used from a command and control centre to attack several high-profile websites. The use of IoT devices allowed the attackers to circumnavigate traditional security measures.

In an IP spoofing attack, an attacker tries to impersonate as an authorised user by crafting a packet with a forged IP address and adjusting certain other fields to make it look legitimate. Having looked at examples of network attacks, we will now examine the security on each layer of the protocol stack.

3 Application-Layer Security

As an example of an application-layer security protocol,Alice and Bob want to use email. In a simplistic scenario, Alice and Bob would decide to use an encryption algorithm such as AES with a 128 or 256-bit key to encrypt their messages. This meets their confidentiality requirement as the message cannot be decrypted by anyone other than Alice and Bob. However, this would require Alice and Bob to agree on a shared key. Distributing this key over the network makes the secret key an easy target for Eve or Mallory. Also, the above scenario fails to provide integrity and origin authentication. The message can be altered as it traverses the network. Alice and Bob (in this instance, their email clients) must use additional measures to provide message integrity and origin authentication. In a variant of this setting, it is also likely that Alice and Bob do not care about the confidentiality of their messages, but they want assurance that their messages will not be tampered with in transit. Alice could calculate the hash of her message using the SHA-3 algorithm and send it to Bob. On receiving this message, Bob would recalculate the hash and verify whether there is a match. However, a potential attacker could easily replace the genuine message with a forged one and a matching hash. Bob cannot tell whether the message sent by Alice has been altered since the hash matches. One possible solution for Alice is to use a pre-negotiated symmetric key to encrypt the hash. Bob now decrypts this hash using the pre-negotiated symmetric key and verifies the integrity of the message received. This also authenticates that the message was sent by someone who shares a key with Bob, in this instance Alice.

We highlighted the challenges of key distribution over the network. See the Cryptography KA for details of public key cryptography. We will ignore the confidentiality requirement for the moment. Alice signs the hash of her message using her private key. Bob then decrypts the message using Alice's public key. This allows for an integrity check and authentication at the same time, as no one other than Alice knows her private key. We avoided pre-negotiation or sharing of keys. So, how does Bob get Alice's public key and trust that Eve or Mallory are not using a forged public/private key to perform MiTM? We provide a brief introduction to key management in the context of public key cryptography in the next section, as it is used by a number of network security protocols. The above example also achieves non-repudiation, as it can be proved that the hash (or in other cases, the whole message) was signed by Alice's private key and she could not deny this fact.

3.1 Public Key Infrastructure

Public key infrastructure (PKI) provides a solution for registering and managing a trustworthy public key. Government agencies or standard organisations appoint or recognise registrars who issue keys, and keep track of the public certificates of entities (individuals, servers, routers etc). The registrars, a large number of which are private companies, themselves have a registered public/private key pair with stakeholders relevant to the application domain. The idea is similar to registering your motor number plate with an authority. Alice generates a pair of public/private keys for herself using her computer. She then presents her proof of identity to one of the registrars. The registrar then issues a certificate to Alice. This certificate is signed by the registrar's private key and can be verified by anyone using the registrar's public key. Typically, a user's identity, public-key and CA information are used as an input to the hash function. The hash is then signed with the CA's private key to produce a Public Key Certificate (PKC). The fields on the certificate include a unique identifier/serial number, a signature algorithm used by the CA and the period of validity. The IETF RFC1422 and ITU-X.509 standards have prescribed the format and standard for managing PKI [4]. Organisations can also manage their own private PKI. CAs also publish a list of revoked certificates which have either expired or been revoked. The web of trust is an alternative scheme where users can create a community of trusted parties by mutually signing certificates without needing a registrar. Continuing with our email example, Alice could send her certificate to Bob along with her email message. Bob is now able to check the validity of the certificate presented by Alice. In our simple example, Alice and Bob could use these techniques to build a secure email system. Pretty Good Privacy (PGP) was one of the earliest email systems to propose the security approach described above, albeit using the web of trust for certificates. Generally, in order for systems to be compatible across platforms and between vendors, application developers make use of the standard application layer protocol, the Simple Mail Transfer Protocol (SMTP) for exchanging messages between mail servers. The content itself is formatted based on a set of standards called Multipurpose Internet Mail Extensions (MIME). As the original Internet protocols lacked security features, a secure version SMIME was developed in order to add an integrity check and certificates to the email header. The functions of the certificate verification and checking revocation list are automatically performed by Alice and Bob's mail agents.

The existing PKI model has faced several challenges, as evidenced by a number of documented cases where Certificate Authorities have issued certificates in error, or under coercion, or through their own infrastructure being attacked. Recent years have seen many partial solutions such as certificate pinning and public immutable logs of issued certificates being implemented to prevent the PKI trust model from being undermined. [5].

3.2 DNS Security Extensions

Internet design philosophy mandates keeping the Internet core functions implemented in the backbone routers to be simple along with other supporting functions to be deployed at the edge. For most people, human cognition means that it is easier to remember host/server names, e.g., cnn.com, over an IP address 151.101.1.XX. Internet routing and other protocols, however, function using IP addresses. The IETF has designed an application-layer protocol, the Domain Name Service (DNS), which performs the translation between a host name and the corresponding IP address. This mapping is performed and maintained by a hierarchy of name servers. There have been a number of DDoS attacks in recent years [6]. We provide an overview of attacks on DNS. In an MiTM, Mallory can impersonate a DNS server, return a bogus address and divert traffic to a malicious server, thus allowing it to collect user passwords and other credentials. A DNS cache poisoning attack, aka DNS spoofing, allows attackers to plant bogus addresses, thus diverting a user request to malicious servers. However, the robust distributed design of the DNS has fortunately saved us from a total collapse of the Internet. Learning from these attacks, the IETF introduced a secure version called DNS Security Extensions (DNSSEC). DNSSEC uses techniques similar to our secure email example above by sending a response signed by the private key of a DNS server. The authenticity of the DNS records is proven by the fact that a responding server signs the record using its private key, which a requester can verify using the corresponding public key. In addition, a digital signature also provides the integrity of the response data. An astute reader may note that confidentiality is not a significant issue for this transaction. More than half a dozen IETF RFCs cover DNSSEC. A study by Chung et al. [7] suggests that only 1% of domains use the DNSSEC mechanisms for security. Very few registrars support DNSSEC and other mechanisms, as communicating DNSSEC information has several security vulnerabilities. DDoS defence is not part of DNSSEC. We will look at defence mechanisms in IDS/IPS in section 8.

3.3 Hyper Text Transfer Protocol Secure (HTTPS)

The most prominent application-layer protocol, the Hyper Text Transfer Protocol (HTTP), was designed without any security considerations. The popularity of HTTP and its wide adoption for e-commerce imposed strict security requirements on this protocol. A secure version called HTTPS was introduced by using security services from the transport layer, which allows the URL, content, forms and cookies to be encrypted during communication. We discuss the secure transport layer protocols in the next section. A new version, HTTP 2.0, has further enriched the security features of HTTP 1.0. Although not mandated, most browsers support confidentiality by encrypting data. New features such as header compression and flow control require servers to maintain additional state information. An attacker could send a large number of empty or tiny frames and keep the server busy processing frame headers. Servers must employ a threshold on the number of connections being processed to limit such attacks.

3.4 Network Time Protocol (NTP) Security

The Network Time Protocol [RFC 5905] is an application-layer protocol used to synchronise devices (hosts, server, routers etc.) to within a few milliseconds of Coordinated Universal Time (UTC). The protocol is typically implemented either as a client-server model or a peer-peer one. In the client-server model, the client sends a request using UDP on port 123 and receives a response back from the server. As with other application-layer protocols, NTP has been subject to replay, DoS and MiTM attacks. Further, an intruder could delay a packet between client server, thus skewing the timing calculations. In a DoS amplification attack, an attacker can send a few bytes of the `MONLIST` command and get the server to send a list of the last 600 clients that made an NTP request. A possible countermeaure would require restricting access to this command from internal hosts only. The most recent implementation of the NTP daemon `ntpd)` uses a hierarchical security model implementing several PKIs, digital signatures and other standard application-layer security mechanisms.

4 Transport-Layer Security

In the previous section, we discussed ways in which applications could build security features by using cryptographic primitives. Data sent over the TCP/IP protocol were not safe and hence each application had to take care of security itself. Ideally, if the transport-layer could provide confidentiality, integrity and authentication mechanisms, the application-layer could be relieved from the burden of security and use the transport layer services instead. This would also provide compatiblity across platforms/vendors. These capabilities are provided by a shim layer between the application and transport layers called the Secure Socket Layer (SSL). A standard application programming interface (API), similar to the socket API, allows applicationsto bootstrap secure connections and to send/receive data securely. IETF started to develop the Transport Layer Security (TLS) borrowing most of its ideas from the SSL 3.0 protocol. The most prominent web browsers have started to support the latest TLS 1.3 standardised in 2018. In this section, our discussions will relate to a simplified version of the security features in order to understand the basics of the TLS protocol. The exact syntax and semantics of the protocols and a rich set of configurations are described in hundreds of pages of RFCs.

We will now bring Alice (server) and Bob (client) back into the action. Alice is configured with her public/private key pairs, as described in 3.1. It is worth emphasising that some of these basic techniques are also used in security protocols on other layers, which we will discuss in this chapter. The TLS protocol has 3 phases: handshake, key-derivation and data transfer, as shown in figure 2.

4.0.1 Handshake

1. First Bob and Alice exchange the three-way TCP SYN, SYNACK and ACK messages. It should be noted that this step is not part of TLS/SSL.

2. Bob then sends a ClientHello message to Alice along with the cipher suites (ciphers and the hash functions it supports) and a nonce, a large, random number, chosen specifically for this run of the protocol.

3. Alice responds with a ServerHello message along with her choice from the cipher suites (e.g., AES for confidentiality, RSA for the public key, SHA2 for the Message Authentication Code

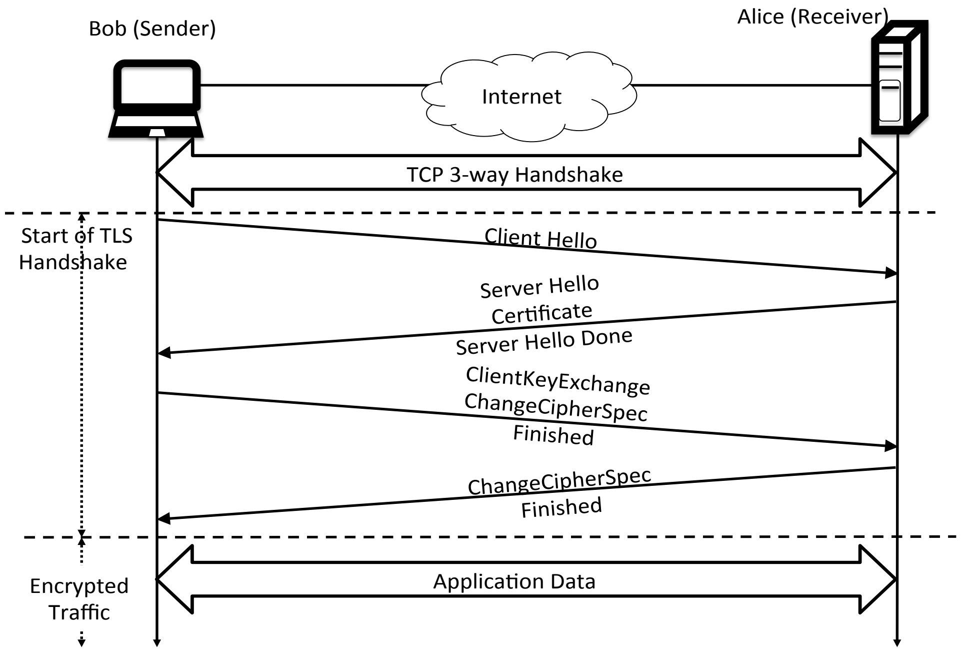

Figure 2: TLS Handshake

(MAC)), a certificate containing her public key and a nonce. Additionally, she could also request the client's certificate and parameters for other TLS extensions.

4. Bob checks validity of the certificate and is assured that it belongs to Alice. He initiates the ClientKeyExchange message. This can use a range of key exchange methods, e.g., RSA or the Diffie-Hellman (and variants) to establish a symmetric key for the ensuing session. For example, when using RSA, Bob could generate a 48-bit Pre-master secret (PMS) and encrypt it with Alice's public key obtained using the steps as described above and send it to Alice.

5. Bob sends a ClientCipttherSpec and a Finished Message suggesting that the key generation and authentication are complete.

6. Alice also has the shared key at this point. She responds with a ChangeCipherSpec and a Finished Message back to Bob.

7. Bob decrypts the message with the negotiated symmetric key and performs a message integrity check.

After successfully completing the above steps, a secure tunnel is established and the encrypted application data can now be sent, as shown at the bottom of figure 2. The details of the protocol exchange and message processing can be found in [4].

4.0.2 Key-Derivation

The client nonce, server nonce and PMS are input into a pseudorandom function to produce a master secret. All the other key data for this connection are derived from this master secret in conjunction with the additional parameters. The following four common keys are derived at both ends:

1. Session encryption key for data sent from Bob to Alice (client encryption key).

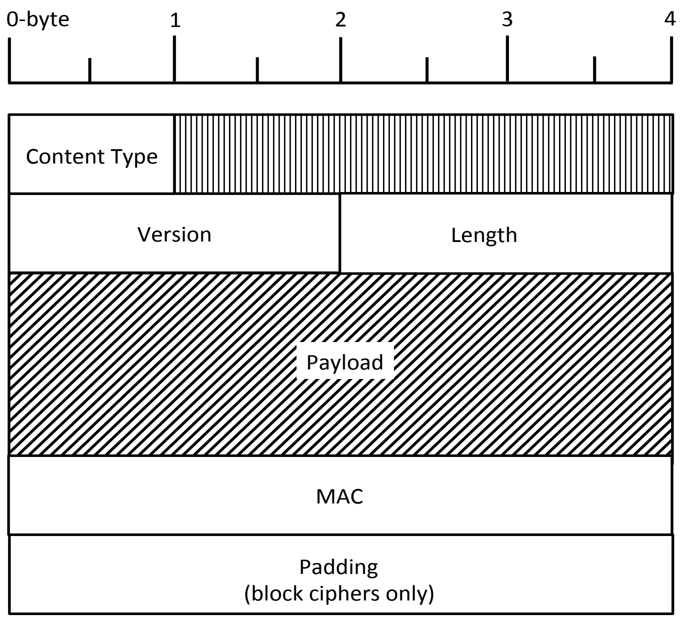

Figure 3: TLS Record Structure

2. Session encryption key for data sent from Alice to Bob (server encryption key).
3. Session MAC key for data sent from Bob to Alice (client MAC key).
4. Session MAC key for data sent from Alice to Bob (server MAC key).

Bob and Alice derive separate keys for encryption and integrity in each direction for enhanced security. Generating these ephemeral keys allows for perfect forward secrecy, as these keys cannot be reused in future sessions. For example, Eve could capture every communication between Alice and Bob. She could pretend to be Bob and repeat the sequence of commands sent by Bob later in the day. This attack is called a *connection replay* attack. The TLS and most other protocols use a session specific nonce, a random number, to avoid this attack. The PMS generation algorithm uses a nonce in the mix. The connection replay attack will fail, as Alice would have a different set of keys from Eve for the new session due to this new nonce.

4.0.3 Data-Transfer

TCP is a byte oriented transport protocol where application-layer data are sent as a stream of bytes. Integrity check algorithms require fixed length data for a MAC calculation. If applications have to collect and pass fixed length data to these algorithms, further delay will be incurred. Hence, TLS defines a record format, as shown in figure 3, where the length of the data sent in each record can be indicated along with the type of record (data or control). A MAC is also appended at the end of each record. For example, if data are sent from Bob to Alice, the session MAC key for the data sent from Bob to Alice are used to generate this MAC. Further, the data plus the MAC are encrypted using the session encryption key for data sent from Bob to Alice.

As the TCP sequence number is not encrypted, a possible MiTM attack could simply capture the TCP segments and swap the TLS records between these segments. A receiver would not be able to detect this attack as the integrity of the TLS records remains unchanged. the TLS provides a separate mechanism where the sender and receiver keep track of the record sequence number without explicitly exchanging it. However, the MAC calculations at both ends use this sequence number in the mix. Any MiTM rearrangement of records will fail an integrity check.

Having discussed the technical details of the TLS, we now consider how it performs in the presence of certain attacks. In a Password Sniffing attack, Eve captures a few packets and wants to get passwords in HTTPS or other application traffic. As the user data are encrypted, the password can not be sniffed. In an IP Spoofing attack, Mallory uses a forged IP addresses to fool Bob into accepting bogus data. Mallory must be in possession of the secret key as well as the forged IP address to succeed. An MiTM attack is prevented by using public key certificates to authenticate the correspondents.

We note that in a related transport-layer attack called a *SYN Flooding DDoS attack*, a group of attacking machines keep sending TCP SYN messages to request a connection and let the server allocate resources. However, this type of attack can be handled by the TCP and hence is not duplicated in the TLS. A defence known as SYN Cookies has been implemented in many operating systems [RFC4987]. The server does not half open a connection right away on receiving a TCP connection request. It selects an initial sequence number (ISN) using a hash function over source and destination IP addresses, port numbers of the SYN segment, as well as a secret number only known to the server. The server then sends the client this ISN, aka Cookie, in the SYNACK message. If the request is from a legitimate sender, the server receives an ACK message with a new sequence number which is ISN plus 1. Once this validation is done, the server opens a TCP connection. A DDoS sender would either not respond with ACK or would not have the correct ISN in its response. Hence, no TCP sources have been wasted.

The current version of SSL (and TLS) has evolved through experiencing several attacks and vulnerabilities found in earlier versions. *SSL Stripping attacks* remove the use of SSL/TLS altogether by modifying unencrypted protocols which request the use of the TLS. The *BEAST attack* exploits the predictable initialisation vector of TLS 1.0 implementation due to use of the Cipher Block Chaining (CBC). This allows an attacker to decrypt parts of a packet, e.g., HTTP cookies. A long list of known attacks and mitigation were discussed in RFC 7457. Many of these vulnerabilities are also attributed to either an improper implementation or poor understanding of the protocol suite rather than a lack of proper specifications. For example, the TLS design problem of calculating MAC before encryption results in a timing side-channel attack called the Lucky Thirteen attack, which allows attackers to decrypt arbitrary ciphertext. Countermeasures for this attack include using AES-GCM enrcyption, or using the encrypt first and then calculating the MAC approach [RFC7366].

4.1 Quick UDP Internet Connections (QUIC)

QUIC is a new transport protocol designed by Google for faster web-browsing using UDP instead of HTTP over TCP. The protocol currently uses proprietary encryption and authentication. Firewalls and IDS systems typically detect HTTP traffic, and perform deep packet inspection, virus scanning and other security measures. Although QUIC uses the standard HTTP ports, security devices do not track this application layer protocol at present. It is treated as regular UDP traffic. Since the standardisation work is already in progress, it is likely to use TLS1.3 for secure transport.

In this section, we looked at various mechanisms for securing the end-to-end communication channel via transport protocols. However, if the content being transferred becomes accessible to an attacker outside the communication channel, they could compare the volume of the encrypted material and make inferences. As a consequence, it could potentially compromise message confidentiality.

5 Network Layer Security

Although application-layer and transport-layer security help to provide end-to-end security, there is also merit in adding security mechanisms onto the network layer. First, higher-layer security mechanisms do not necessarily protect an organisation's internal network links from malicious traffic. If and when malicious traffic is detected at the end-hosts, it is too late, as the bandwidth has already been consumed. The second major issue is that the higher-layer security mechanisms described earlier (e.g., TLS) do not conceal IP headers. This makes the IP addresses of the communicating end-hosts visible to eavesdroppers.

Additionally, many organisations prefer their traffic to be fully encrypted as it leaves their network. In the early days of networking, several private networks were in use. However, maintaining a private network may not be cost effective. An alternative solution is to make use of the Internet to connect several islands of private networks owned by an organisation. Also, employers and employees want a flexible work environment where people can work from home, or connect from a hotel room or an airport lounge without compromising their security. We have already determined that the Internet is unsafe. The concept of a virtual private network (VPN) over the public Internet requires a set of network layer security mechanisms that we will explore in this section. We start our discussion with security additions to the network layer IP protocol called IPsec. Figure 4 shows that an employee working from home accesses a server at work, the VPN client in their host encapsulates IPv4 datagrams into IPsec and encrpyts IPv4 payload containing TCP or UDP segments, or other control messages. The corporate gateway detects the IPSec datagram, decrypts it and decapsulates it back to the IPv4 datagram before forwarding it to the server. Every response from the server is also encrypted by the gateway. We note that encryption is not mandatory in IPsec. Figure 4 is one of several modes of operation for IPsec. For example, there could be two corporate networks, each with their own IPsec gateway communicating over the open Internet.

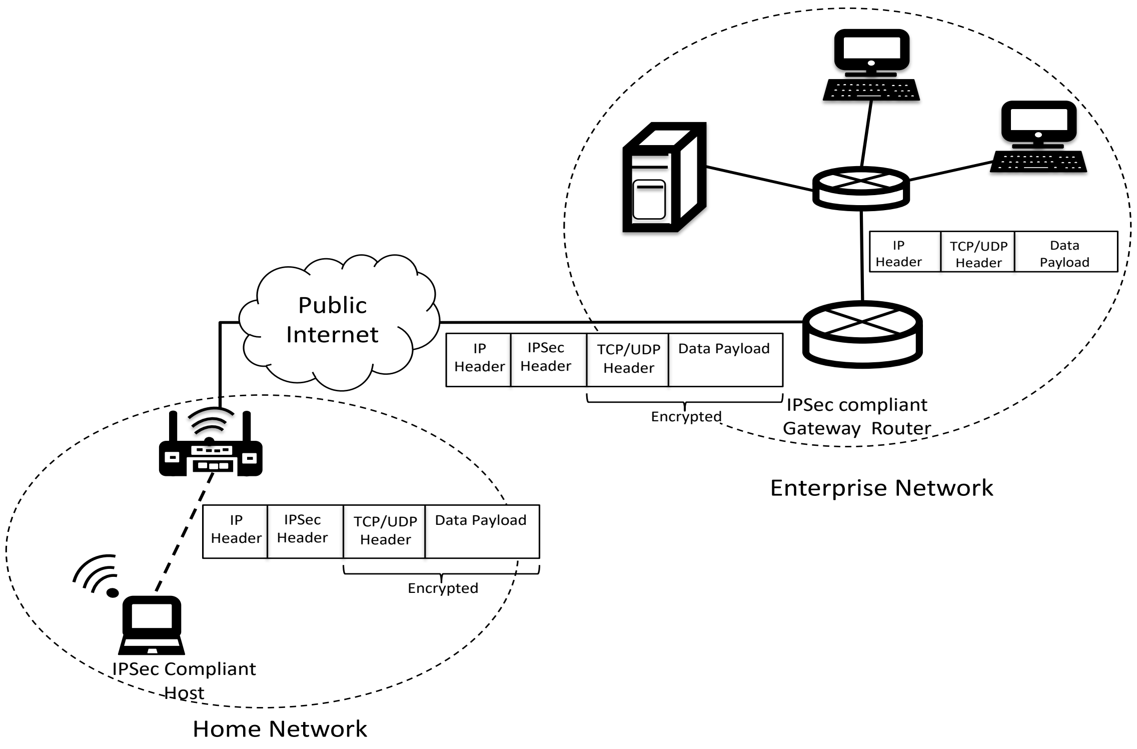

Figure 4: Example IPsec Client Server Interaction

We started off with a simple example showing data confidentiality using encryption. However, IPsec also provides data integrity, origin authentication and replay attack prevention. Again, the set of modes/configurations/standards provided by IPsec is extensive; interested readers should access the relevant IETF RFCs for formats and protocol details.

IPsec supports Tunneling and Transport modes of operation. In Transport mode, as shown in figure

5, the original IP header is used but the rest of the payload gets encrypted. In our example of figure4, if transport mode is used, it would require a routable IPv4 address. This can be achieved if the endpoint is behind a NAT. Details of NAT traversal can be found in RFC7296.

In the rest of this section, we will discuss the widely used alternate Tunneling mode in detail. If the edge devices (routers/gateways) of two networks are IPsec aware, the rest of the servers/hosts need not worry about IPsec. The edge devices perform the encapsulation of every IP including the header. This virtually creates a secure tunnel between the two edge devices. The receiving edge device then decapsulates the IPv4 datagram and forwards within its network using standard IP forwarding. Other possible configurations for a tunnel could involve one IPsec aware host and an IPsec aware gateway (as in figure 4). A tunnel between two IPsec aware hosts is also possible without involving edge routers. The Tunneling mode remains in widespread use due to its simplicity, as it does not require IPsec protocol support in the end hosts. Also, key negotiation is simplified, as two edge devices can handle connections on behalf of multiple hosts in their respective networks. An additional advantage is that everything, including the IP source/destination address, gets encrypted, thus making traffic analysis harder. The ESPv3 allows to use the Traffic Flow Confidentiality (TFC) mechanisms which adds arbitrary length padding to obfuscate the traffic pattern and prevent avoid statistical traffic analysis attacks. Kiral et al. [8] reported experimental results exploring padding and several other techniques such as packet framgmentation, introduction of artificial inter-packet delay, inserting of dummy packets to avoid traffic analysis.

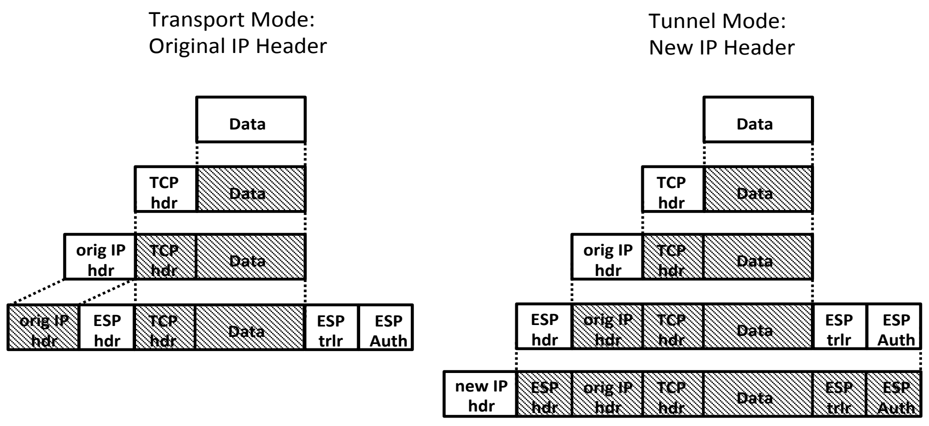

Figure 5: Transport and Tunnel Mode Encapsulation

IPsec supports a set of formats to implement security. The Encapsulation Security Payload (ESP) format supports confidentiality using encrypted IP packets, data integrity using hash functions, and source authentication. If an application does not require confidentiality, it may simply use the Authentication Header (AH) format, which supports data integrity and source authentication. The IETF RFC2410 defines the NULL Encryption algorithm with ESP to achieve the same outcome. In total, we get four different options for communication: Transport mode with ESP, Transport mode with AH, Tunnel mode with ESP and Tunnel Mode with AH. Since VPN tunnels are fully encrypted, the Tunnel mode with ESP remains the protocol of choice.

Two entities participating in IPsec communication establish *security association (SA)* for each direction of the link. Essentially, a number of variables are recorded in a database called the Security Association Database (SAD) for lookup during IPsec protocol processing, somewhat similar to the TCP connection state. Some of the state information includes the type of encryption used (e.g., AES or 3DES), the encryption key, the type of integrity check used (e.g., SHA-2 or MD5), the authentication key and so on. An Anti-Replay Window is also used to determine whether an inbound AH or ESP packet is a replay.

When a large number of end-points use IPsec, distributing the keys becomes challenging. The RFC7296 defines the Internet Key Exchange protocol (IKEv2). Readers will observe a similarity between TLS 4 and IKE, in that IKE also requires an initial handshake process to negotiate cryptographic algorithms and other values such as nonces and exhange identities and certificates. We will skip the details of a complex two-phase protocol exchange which results in the establishment of a quantity called SKEYSEED. These SKEYSEEDs are used to generate the keys used during a session, as we recall IPsec SAs. We note that the IKEv2 has evolved from IKEv1, Internet Security Association and Key Management Protocol (ISAKMP), and several other earlier efforts. The Internet Security Association and Key Management Protocol (ISAKMP) is a framework that defines the procedures for authenticating the communicating peer, creation and management of Security Associations, and the key generation techniques. It can also provide threat mitigation against DoS and replay attack. Defined in RFC 2408, ISAKMP is also part of IKEv2 for key exchange.

5.1 IP Masquerading

Due to the shortage of IPv4 address space, Network Address Translation (NAT) was designed so that private IP addresses could be mapped onto an externally routable IP address by the NAT device [1]. For an outgoing IP packet, The NAT device changes the private source IP address to a public IP address of the outgoing link. As a consequence, it obfuscates the internal IP address from the outside world. To a potential attacker, the packets appear to be coming from the NAT device, not the real host/server behind the NAT device.

5.2 IPv6 Security

Our discussions about security so far have assumed the use of IPv4. The shortage of IPv4 addresses resulted in the development of new IPv6 protocol, as the NAT mechanism had several flaws. As IPv6 adoption is gradually increasing, we should highlight the security benefits and challenges associated with the deployment of IPv6. For example, the use of 128-bit address space means attackers need a lot more time to scan the ports, as opposed to IPv4, where the entire address space can be scanned in a few hours. Several security problems associated with ARP, which can be discussed later in this chapter, disappear as IPv6 layer-3 addresses are derived directly from layer-2 addresses without any need for address resolution. However, this allows attackers to infer information about the host/servers which can be handy when launching attacks. Using hash function for address generation is recommended as a mitigation technique. Further, IPv6 allows for a cryptographically generated address (CGA) where an address is bound to a public signature key. This helps when authenticating between routers for secure message exchange. Initially, IPsec was mandated to be used in IPv6 networks, but due to implementation difficulties it remains a recommendation. Several informational IETF RFCs and vendor-specific white papers provide a comprehensive treatment of IPv6 security challenges.

5.3 Routing Protocol Security

So far, we have primarily focussed on the security of data being sent using the TCP/IP protocol suite. However, a network can easily be disrupted if either the routers themselves are compromised or they accept spurious routing exchange messages from malicious actors. First, we will discuss the Interior Gateway Protocols (IGP) which are used for exchanging routing information within an Autonomous

System (AS), an ISP, for example. Two prominent protocols: Routing Information Protocol (RIPv2) and Open Shortest Path First (OSPFv2) are in widespread use with ASes for IPv4 networks. The newer RIPng and OSPFv3 versions support IPv6. These protocols support no security by default but can be configured to support either plain text-based authentication or MD5-based authentication. Plain text authentication sends a secret key in clear text along with routing updates, thus making it easy to be sniffed by a packet analyser. A more secure option uses the MD5. Routers exchange a message digest and a key-id along with the routing updates. The Key-id indicates which key to use from a list of passwords. This avoids the sniffer attack. Authentication can avoid several kinds of attacks such as bogus route insertion or modifying and adding a rogue neighbour. Additionally, routers may employ route filtering to avoid propagating the only legitimate route.

5.3.1 Border Gateway Protocol (BGP) Security

The Internet uses a hierarchical system where each AS managed by an ISP, exchanges routing information with other ASes using the Border Gateway Protocol (BGP). See RFC1163 and RFC1267 for the details of the BGP protocol. After receiving an IP prefix [1], the reachability information, from its neighbour, the router checks the newly received information against its stored knowledge to see if there is a better path to reach a destination network. This information is updated locally and propagated to its immediate neighbours. The distributed system allows networks to reach each other globally.

In recent years, attacks on the BGP have been seen with the apparent intention of disrupting YouTube services globally. The entire Internet experienced an outage in another country due to either misconfiguration or the malicious advertising of bogus BGP updates. Either way, this highlights the security weakness in the BGP protocol. This vulnerability arises because of a lack of integrity and authentication for BGP messages. We will describe some well-known attacks on the BGP protocol.

In what is known as a *BGP route hijacking attack*, a malicious router could advertise an IP Prefix, saying that the best route to a service is through its network. Once the traffic starts to flow through its network, it will then choose to drop all the packets to this service for a variety of reasons, including censorship. It could also read all of the un-encrypted packets. Additionally. the attacker could divert traffic through an unsuspecting AS, thus suddenly increasing their load. In a *BGP denial-of-service (DoS) attack*, a malicious router would send an avalanche of BGP traffic to a victim AS, while keeping its border router busy so that it could not process any valid updates. The attacker could also propagate spurious BGP updates and corrupt routing tables so as to prevent traffic from reaching its intended destination.

IETF is currently working on a standard called BGPSec to address these security concerns. This work is based on a prior proposal called S-BGP [9]. The core of the scheme lies in the use of PKI to verify the signatures of the neighbours sending the updates. Two neighbouring routers could use IPsec mechanisms for point-point security to exchange updates. We will now look at a simple example where a BGP router receives a path ZZZ YYY XXX. The BGP router verifies the signature of AS XXX using PKI mechanisms that we learnt about earlier. It then verifies the signature generated by YYY and subsequently by ZZZ. This allows us to verify the origin and authenticity of the whole chain of updates. However, this approach entails large overheads. Signature verification comes at a cost, implementing BGPSec would require the border routers to verify a larger number of signatures on booting. Additional crypto hardware and memory would certainly help keep the performance on track.

Despite these BGPSec and other standardisation efforts, not many routers deploy these mechanisms due to additional costs and a lack of short-term benefits unless there is a consensus to mandate it globally [10]. Mechanism costs are an additional but smaller barrier to widespread deployment. The existing BGP security proposals suffer from a classic economic problem. A new BGPSec deployment mostly benefits (non-deploying) operators other than those deploying the mechanism; thus the deployer's reward lies in the future, while the losses from non-deploying networks are stacked upfront.

6 Link Layer Security

In this section, we are confining our attention to the security of link layer technologies which are relevant to end-user/PC deployments. Other link layer technologies are addressed in other knowledge areas. We will start our discussion with the prominent 802.1X Port-based Authentication followed by link layer security issues in Ethernet Switched LAN and Wireless LAN environments.

6.1 IEEE 802.1X Port-based Authentication

The IEEE 802.1X is a port-based authentication for securing both wired and wireless networks. Before a user can access a network at the link layer, it must authenticate the switch or access point (AP) they are attempting to connect to, either physically or via a wireless channel. As with most standards bodies, this group has its own jargon. Figure 6 shows a typical 802.1X setup. A user is called a supplicant and a switch or AP is called an authenticator. The architecture requires an Authentication Server (AuthS) that can be implemented using one of the existing protocols: Remote access dial-in user service (RADIUS), DIAMETER, Kerberos, Lightweight Directory Access Protocol (LDAP) or Active Directory (AD), among others. The AS function can also be co-located with the authenticator. We will now consider RADIUS as our example AS. Typically, the AS and authenticator are pre-configured with a shared secret. Using the RADIUS protocol as an example, once a supplicant request is received, the authenticator sends a RADIUS Access Request message to the RADIUS server, requesting authorisation to be granted to access the resources.

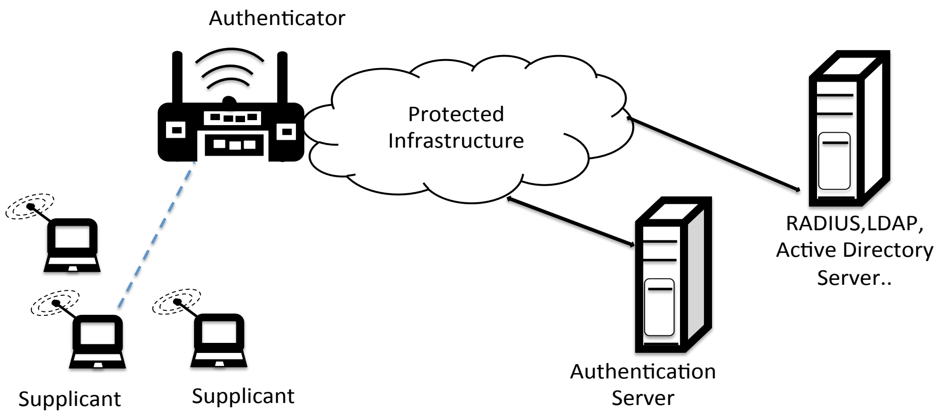

Figure 6: 802.1X Port-Based Authentication Architecture

Supplicant software is typically available on various OS platforms or it can also be provided by chipset vendors. A supplicant (client) wishing to access a network must use the Extensible Authentication Protocol (EAP) to connect to the AS via an authenticator.

6.1.1 Extensible Authentication Protocol (EAP)

The EAP is an end-end client to authentication server protocol. From supplicant to authenticator, it is sent over Layer2 protocols, i.e., EAP over LAN (EAPOL). There is no need for higher layer protocols. As the authenticator is connected to the AS using a trusted link with a shared secret, a higher layer protocol such as RADIUS/DIAMETER over UDP can be used on this side of the link.

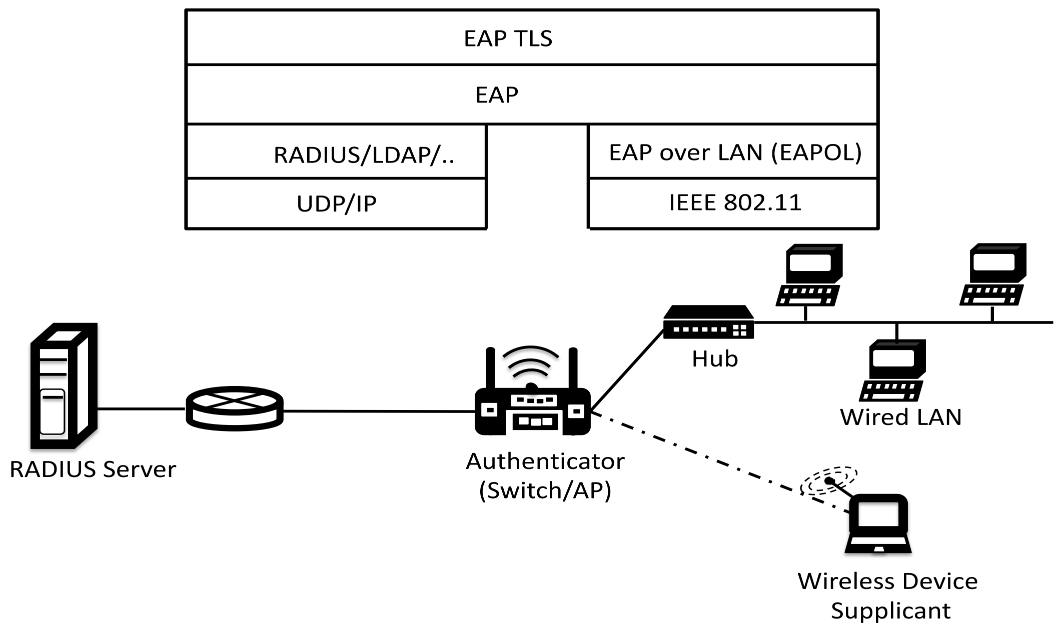

Figure 7: Extensible Authentication Protocol (EAP)

When a new client (supplicant) is connected to an authenticator, the port on the authenticator is set to the 'unauthorised' state, allowing only 802.1X traffic. Other higher layer traffic, such as TCP/UDP is blocked. The authenticator sends out the EAP-Request identity to the supplicant. The supplicant responds with the EAP-response packet, which is forwarded to the AS. This typically includes the supplicant's credentials (username and hash of password). Upon verification, the AS returns one of the following responses: Access Accept, Access Reject, Access Challenge for extra credentials. If the result is Access Accept, the authenticator unblocks the port to let higher layer traffic through. When the supplicant logs off, the EAP-logoff to the authenticator sets the port to block all non-EAP traffic.

Sending a supplicant's credentials in plaintext is problematic for several reasons. To safeguard against any eavesdropping, the EAP uses a Tunnel for authentication and authorisation. A whole range of EAP Tunneling protocols are available. EAP-Transport Layer Security (EAP-TLS), EAP for GSM Subscriber Identity (EAP-SIM) and EAP Protected Authentication Protocol (EAP-PEAP) are some of the examples for establishing a secure tunnel. EAP-PEAP, also known as 'EAP inside EAP' is one of the most popular protocols. If we dig deeper, before the port is unblocked, a complex process is used to generate a dynamic encryption key using a 4-way handshake. Essentially, all of these protocols establish a TLS tunnel but differ in choice of hash algorithms, the type of credentials used, whether a client-side certificate is used etc. Most protocols would use a server side certificate.

Once the supplicant and AS mutually authenticate, they together generate a Master Key (MK). As we have already discussed, the authenticator has been playing the role of a relay up to this point. During this process, the supplicant derives a Pairwise Master Key (PMK). The AS also derives the same PMK and sends this to the authenticator. From this point on, the supplicant and authenticator use the PMK to derive the Temporal Key (TK) used for the message encryption and integrity. The key derivation process is similar to what we learnt in the TLS earlier. We will revisit key generation and the relationship between the various keys in detail later in the Robust Secure Networking (RSN) section.

6.2 Attack On Ethernet Switch

Although the research literature has primarily focused on higher layer security, the Stuxnet [11] attack has demonstrated that an innocuous looking USB drive could easily wreak havoc in a local area network (LAN) environment without any need for an Internet connection. Widely deployed Ethernet technology is built around self-learning and configuring protocols. This allows for ease of management, but at the same time introduces several security vulnerabilities [12]. We provide a brief review of some of the possible attacks here.

Media Access Control Attack: Switch Poisoning Attack

Ethernet switches keep forwarding table entries in a Content Addressable Memory (CAM). As a switch learns about a new destination host, it updates the table and for all future communications, this table entry is looked up to forward a frame. Unlike broadcast Ethernet or Wireless LAN, these frames are not accessible to hosts attached to other ports. However, if the switch does not have a mapping to a new media access control (MAC) address, i.e., which port to forward a new frame to, it will flood the frame on all of its outgoing ports. An attacker could craft several frames with random addresses to populate an entire CAM. This would result in the switch flooding all the incoming data frames to all the outgoing ports, as there is no space available to enter a new mapping. This makes the frame available to the attacker attached to one of these ports. As a consequence, a MAC flooding attack would also affect all the VLANs filling their CAM. However, this kind of attack requires an attacker to control a device that is directly connected to an Ethernet switch or possibly to some used but unattended Ethernet wall sockets which are still connected to a port. Mitigating this kind of attack would require authenticating and verifying the MAC addresses from some local database of legitimate addresses before populating the forwarding table entry.

MAC Spoofing: attacks occur when an attacker eavesdrops on a link and detects the MAC address of a target host. It then masquerades as a legitimate host by altering its host's MAC address to match the newly detected MAC address. The attacker floods the network with the newly configured MAC address while directing the traffic to itself by altering the switch forwarding table entry. The switch is now tricked into forwarding the frames destined for the target host to the attacking host.

The MAC address is not not designed or intended to be used for security. The 802.1X, which we discussed earlier, is a good starting point for preventing unauthorised users from accessing any service on a network. As a side issue, a user may choose to spoof his or her MAC address in order to protect his or her privacy. Most popular operating systems support MAC address randomisation to avoid devices being tracked based on a MAC address.

Address Resolution Protocol (ARP) Spoofing: attacks occur when an attacker sends a fake ARP message over a LAN, binding the target's IP address to its own MAC address. Once it manages to compromise the ARP table, it will start receiving any data that were intended for the target's IP address. ARP spoofing can also be used for DoS attacks by populating the ARP table with multiple IP addresses corresponding to a single MAC address of a target server, for example. This would then redirect unnecessary traffic to the target, keeping it busy processing these messages. ARP Spoofing is also helpful in session hijacking and MiTM attacks.

In fact, a mitigation scheme would set limits on the number of addresses that can be learnt per-port on a switch. Some vendors use a verification process where they inspect the MAC address and IP address information in ARP packets against the MAC-IP bindings contained in a trusted binding table.

This allows for any ARP packets that do not have an entry in the binding table to be discarded. The binding table must be updated frequently to avoid blocking legitimate updates.

VLAN hopping: VLAN hopping attacks allow an attacking host on a VLAN to gain access to resources on other VLANs that would normally be restricted. There are two primary methods of VLAN hopping: switch spoofing and double tagging.

In a switch spoofing attack, an attacking host impersonates a trunking switch responding to the tagging and trunking protocols (e.g., IEEE 802.1Q or Dynamic Trunking Protocol) typically used in a VLAN environment. The attacker now succeeds in accessing traffic for multiple VLANs. Vendors mitigate these attacks by proper switch configuration. For example, the ports are assigned a trunking role explicitly and the others are configured as access ports only. Also, any automatic trunk negotiation protocol can be disabled. In a double tagging attack, an attacker succeeds in sending its frame to more than one VLAN by inserting two VLAN tags to a frame it transmits. However, this attack does not allow them to receive a response. Again, vendors provide recommended configuration methods to deal with these possible attacks. A comprehensive survey of Ethernet attacks and defence can be found in [12] and vendor-specific courses.

7 Wireless LAN Security

Wireless LANs are more vulnerable to security risks due to the broadcast nature of media, which simplifies eavesdropping. The Wireless Equivalent Privacy (WEP) protocol, despite being obsolete due to its design flaws, provides several important lessons about how not to design a security protocol. The WEP protocol was designed to provide integrity, confidentiality and authentication. It uses a symmetric key encryption method where the host shares a key with an access point using out of band methods, mostly pre-installation by an administrator or a home network user. The sender calculates a 32-bit Integrity Check Value (ICV) using a cyclic redundancy check (CRC) algorithm over the payload. A 104-bit shared key combined with a 24-bit initialisation vector (IV) is fed into a pseudo random number generator (PRNG) such as a RC4 stream cipher. The plaintext payload and the CRC of the frame are then combined with the key sequence generated by the RC4 using bit-wise exclusive-or operation to encrypt the frame. A new IV is used for each frame.

For authentication, the access points (APs) advertise via beacon frames whether authentication is necessary or not. However, not all APs support this feature. If authentication is required, before association a host connecting to an AP would receive a 128-bit nonce from the AP. It would encrypt the nonce with the shared key and send it back to the AP. The AP would decrypt this response with the shared key and verify whether it matched the nonce it sent originally. The receiver would extract the IV received in plaintext, input IV and shared secret key into PRNG, get a keystream, XOR the keystream with the encrypted data to decrypt data + ICV and finally verify the integrity of the data with the ICV.

The WEP protocol has a number of design flaws. First, the use of a 24-bit IV introduces a weakness into the scheme in that 2^{24} or 16 million unique IVs can be exhausted in high-speed links in less than 2 hours. Given that IVs are sent in plaintext, an eavesdropper can easily detect this reuse and mount a known plaintext attack. Using the RC4 in WEP allows for the Fluhrer, Martin and Shamir (FMS) attacks. In the FMS, an attacker can recover the key in an RC4 encrypted stream by capturing a large number of messages in that stream [13]. The linear CRC algorithm is good for detecting random link errors but is a poor choice for maliciously modifying the message. Strong cryptographic techniques such as message authentication codes and signatures, as discussed in higher layer protocols, are better suited for this task.

Given the poor security design of WEP, the Wi-Fi Alliance took on the job of securing wireless networks. An interim standard called the Wi-Fi Protected Access (WPA) was quickly developed for backward hardware compatibility, while WPA2 was being worked out. WPA uses the Temporal Key Integrity Protocol (TKIP) but maintains RC4 for compatibility. The pre-shared key (PSK), also known as WPA-Personal, is similar to the WEP-Key. However, the PSK is used differently, a nonce, and PSK are hashed to generate a temporal key. Following this, a cryptographic mixing function is used to combine this temporal key, the TA (transmitter MAC address), and the sequence counter resulting in one key for encryption (128 bits) and another key for integrity (64 bits). As a consequence, every packet is encrypted with a unique encryption key to avoid FMS-style attacks. Also, the WPA extends the WEP IV to 48 bits, which is used as a packet sequence counter. It would take 100 years to replay the same IV. A packet received out of order, would be dropped by the receiving station. Several new fields include a new Frame Check Sequence (FCS) field, a CRC-32 checksum for error correction and a hash function based on the new field Michael (MIC) for an integrity check. The WPA has had its own share of attacks, as reported in the literature [14].

The Wifi alliance WPA2 standards derived from the IEEE 802.11i standards were finalised in 2004. WPA2 relies on more powerful hardware supporting a 128-bit AES Counter Mode with the Cipher Block Chaining Message Authentication Code Protocol (CCMP). These methods are discussed in the Cryptography chapter. It also provides an improved 4-way handshake and temporary key generation method.

In 2018, a new WPA3 standard was accepted to make a gradual transition and eventually replace the WPA2. The WPA3 overcomes the lack of perfect forward secrecy in WPA and WPA2. The PSK is replaced with a new key distribution called the simultaneous authentication of equals (SAE) based on the IETF Dragonfly key exchange. The WPA3-Personal mode uses a 128-bit encryption, whereas the WPA3-Enterprise uses 192-bit encryption.

7.1 Robust Security Network (RSN)

An earlier section described the evolution of the WEP into the WPA and WPA2. However, the IEEE 802.11i working group came up with the RSN framework to provide the strongest form of security. It adopts the 802.1X-based mechanisms for access control, as discussed above. Authentication and key-generation are done via the EAP. It continues to use the TKIP and CCMP for various cryptographic functions such as encryption/decryption, integrity check, as well as origin authentication and replay attack detection. Stallings [4] provides a good overview of the RSN protocols and standards.

The RSN Key derivation mechanisms are involved to a degree, as can be seen in figure 8. We will provide a summary of this, as many other protocols (including Cellular GSM) following a similar scheme to the pairwise key scheme provide a mechanism for generating dynamic session keys each time a user starts a new session.

As a starting point, the user device and AP would have a pre-shared key (PSK) using out-of-band methods. However, this is not a scalable solution and in an enterprise setup using the IEEE 802.1X, a Master session key (MSK) is typically generated during the authentication phase. With these two options available, a Pairwise master key (PMK) can be generated in the following two ways: using the PSK as the PMK or deriving the PMK from the MSK using the Pseudo Random Function (PRF). The PSK also uses the host and AP addresses when generating the PTK, thus providing additional defence against session hijacking and impersonation. Further, a nonce is used in the mix to achieve good random keying material. The PTK is now split three ways, thus generating separate keys for each function.

The RSN also caters for a group key generation where a host can communicate with a multicast group, as shown in figure 9. This key is generated by the AP and distributed securely to the hosts associated using the secure pairwise keys derived above. This group key can be changed periodically based on a variety of network policies. The Group Temporal Key generation method is not defined in the standards.

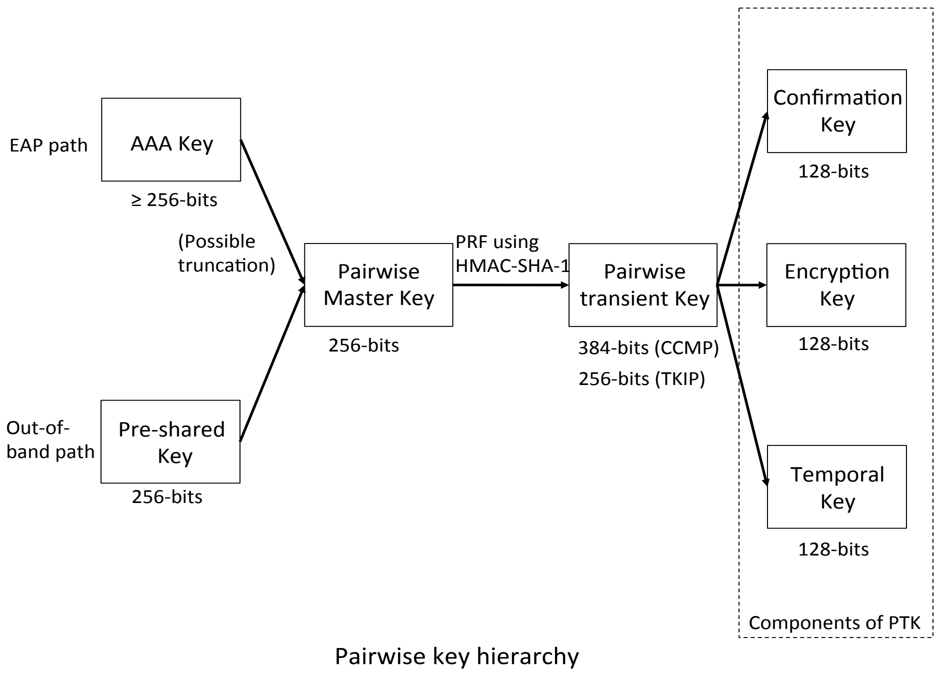

Figure 8: WLAN RSN Pairwise Key Hierarchy

8 Network Defence Tools

Ideally, attacks should be detected as early as possible, or even predicted before they have started so that they can be prevented altogether. We will discuss a number of approaches that can be implemented on various layers of the protocol stack. We provide a brief overview here. The effective deployment of these tools is covered in detail in the Security Operations and Incidence Management KA.

8.1 Packet Filters/Firewalls

The term filter is used for a set of rules configured by an administrator to inspect a packet and perform a matching action, e.g., let the packet through, drop the packet, drop and generate a notification to the sender via ICMP messages. Packets may be filtered according to their source and destination network addresses, protocol type (TCP, UDP, ICMP), TCP or UDP source/destination port numbers, TCP Flag bits (SYN/ACK), rules for traffic from a host or leaving the network via a particular interface and so on. This was typical of the early packet filters, which worked on inspecting header fields. These filters did not retain any state information about the packets/flows/sessions they belonged to. As more computing power and cheaper memory became available, the next generation of packet filters started to track transport layer flow, a chain of packets belonging to a session, known as *stateful* filters.

The packet filters, aka, Firewall system can be co-located with routers or implemented as specialised servers. In either case, they are gatekeepers, inspecting all incoming/outgoing traffic. The filters are set based on a network's security policy and the packets are treated accordingly. Although firewalls play a key role in securing a network, taking down a firewall can potentially wreak havoc for organisations which are dependent on networking technology.

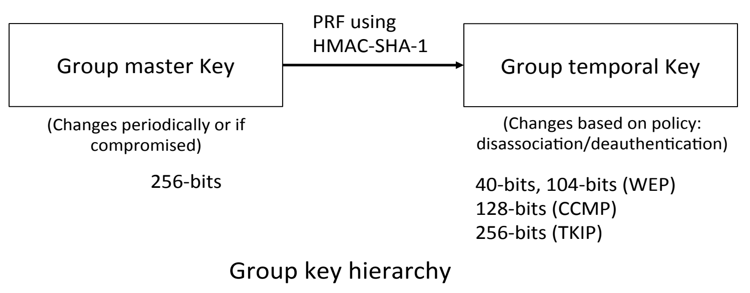

Figure 9: WLAN RSN Group Key Hierarchy

8.2 Application Gateway (AG)

As we saw earlier, a firewall can check rules based on the filtering criterion using TCP/UDP protocol headers, port numbers etc. However, many organisations use application level gateways, aka application proxy, to perform access control, as they facilitate any additional requirements of user authentication before a session is admitted. These AGs can inspect information from the full 5-layer (Internet) or 7-layer OSI stack, except for encrypted bits. In a typical setting, the application gateway will use a firewall's services after performing authentication and policy enforcement. Both the AG and firewall are also co-located in many deployments. A client wanting to access an external service would connect to the AG first. The AG would prompt him or her for authentication before initiating a session to the external server. The AG would now establish the connection with the destination acting as a relay on behalf of the client, essentially creating two sessions: one between the client and the AG, and one between the AG and the destination.

Another interesting application of an AG is SSL termination. An incoming webserver SSL connection could be terminated at the AG, so that it could do the resource intensive encryption/decryption and pass the un-encrypted traffic to the back-end servers. This allows the workload on these busy servers to be reduced in addition to implementing security measures. In practice, the AGs are also configured to inspect encrypted outbound traffic where the clients are configured with corresponding certificates installed at the AG.

Higher level security provided by an AG comes at the expense of additional hardware/software resources. Further, an AG can slow down the connection, as authentication, policy checks and state maintenance are performed to keep track of every session going through the AG. Another complexity involved with an AG is the need to configure it for each application, or possibly be implemented as multiple application specific servers.

8.3 Circuit-level Gateway (CG)

A circuit-level gateway is a proxy that functions as a relay for TCP connections, thus allowing hosts from a corporate Intranet to make TCP connections over the Internet. CGs are typically co-located with a firewall. The most widely used CG today is SOCKS. For end user applications, it runs transparently as long as the hosts are configured to use SOCKS in place of a standard socket interface. A CG is simple to implement compared to an AG, as it does not need to understand application layer protocols.

8.4 Intrusion Detection Systems (IDS)

Intrusion detection systems can provide valuable information about anomalous network behaviour. However, other complementary techniques are also required if all traffic is encrypted. Similar to AGs, they inspect higher layer information and many more attributes of sessions beyond what a packet-filter or firewall can do. An IDS would monitor network traffic with the help of agents/sensors/monitors on the network and sets off alarms when it detects (or thinks it has) suspicious activity. Essentially, the IDS would compare the traffic against what it considers normal traffic and, using a range of techniques, would generate an alert. False alarms are a huge problem for network/security administrators despite decades of research. For example, false positives may be generated by the IDS for legitimate hosts carrying out identical legitimate behaviour that may appear malicious. We will now consider a situation where a legitimate domain accessed frequently by hosts in a network becomes temporarily unreachable. The failed DNS queries to the same domain in this instance would be generated for many hosts and may appear suspicious, but should not be considered malicious activity. Likewise, a false negative would cause classifying malicious activity as benign.

The following are the two main IDS categories:

- Signature-based intrusion detection systems compare monitored traffic against a database containing known threat signatures similar to virus scan software. The database has to be continually updated, however, or it will not detect new types of attacks. Signatures can be as simple as a source/destination IP address or contain many other protocol headers including certain patterns in the payload. We provide a simple example from an open source IDS Snort below.

  ```
  alert tcp any any -> 192.168.5.7/24 80
  (content:"GET"; msg:"WWW GET has been detected";
  sid:1000007; rev:1;)
  ```

  In this simple example, the action is 'alert'. The source is defined for any TCP flow with any address. The destination is defined as 192.168.5.7/24 at port 80. The rule is defined to check whether the packet contains a 'GET' string and then generate an alert. The 'sid' or Snort Identifier refers to the Snort rule used. Snort provides a long set of rules but allows users to define their own.

  IDS generates a heavy workload, as it has to compare huge numbers of signatures. Speed of detection plays a key role in preventing these attacks. Several systems deploy parallel and distributed detection systems that can cope with high traffic rates on large networks and allow online detection; others exploit parallelism at the hardware level in order to overcome processing delays so that packets and flows can be processed at high speeds, thus providing faster results. A lot of research has also focused on faster patterns or rule matching with the aim of reducing packet processing delays.

- Anomaly-based intrusion detection systems use statistical features of normal traffic to compare with the monitored traffic. The criterion for capturing normal traffic could be bandwidth usage, protocols, ports, arrival rate and burstiness [15]. For example, a large percentage of port scans would generate an alert. Attacks can be detected by monitoring hosts or networks for behaviour typical of different attacks. A target link flooding attack aims to overwhelm a particular link in the network, thus disconnecting a selected network region or server from the network. Observing an increase in `traceroute` packets in the network could indicate an upcoming target link flooding DDoS attack.

Despite using machine learning techniques such as Linear Regression, Neural Networks, Deep Learning etc., which train a classifier from normal or malicious data and use it to identify the same behaviour within future data, these systems' false positives remain high [16].

Another way of classifying IDSes is the point of monitoring for malicious behaviour. A host intrusion detection system (HIDS) runs on individual hosts in the network. Most virus scan software would have this feature where they also monitor inbound and outbound traffic in addition to the usual virus scanning. This can be particularly helpful if the hosts have been compromised and form part of a bot to attack other servers/networks. In contrast, a network intrusion detection system is deployed at strategic locations within the network to monitor inbound and outbound traffic to and from the devices in various segments of the network.

8.5 An Intrusion Prevention System (IPS)

An intrusion prevention system distinguishes itself from an IDS in that it can be configured to block potential threats by setting filtering criteria on routers/switches at various locations in the network.

IPS systems monitor traffic in real time dropping any suspected malicious packets, blocking traffic from malicious source addresses or resetting suspect connections. In most cases, an IPS would also have IDS capabilities. Several IDS/IPS tools generate an alert for a spam preparation stage, which is indicated by a rise in DNS MX queries that spam bots generate to discover a mail server before sending spam emails.

The IPS system is proactive and, in theory, it should work autonomously without intervention from a security/network administrator. For example, on inspecting the headers, if an IPS system suspects an email to be unsafe, it could prevent it from being forwarded to an end user. However, the risk of blocking legitimate traffic is a huge problem due to false positives or the mis-configuration of these systems. In practice, however, IPS systems are mostly set to detect modes and start blocking traffic only when the confidence in the incidence being true positive becomes high.

IDS/IPS vendors provide regular signature updates and security teams will have to determine which ones to deploy, depending on the network environment that it is deployed. IDS/IPS can also be software, deployed on the application layer on strategic endpoints. These do not have their own OS, relying instead on the host, but can be fine-tuned to support and protect the specific device it is deployed to.

There are several other mechanisms for network defence. In highly secured environments such as defence or critical infrastructure, a device known as a Data Diode can be configured to allow a secure flow of data in one direction only. For example, a water dam could provide information on water levels to people living in the neighbourhood, but may restrict sending any information back to the dam control network. A comprehensive coverage of this topic can be found in the Security Operations and Incidence Management KA.

8.6 Network Architecture Design

These network protection tools are most effective when deployed in combination, where different local networks have distinct and focussed purposes. Network design must balance the concerns of cost and performance against the benefits of segmenting traffic as much as possible. An early example was Network Perimeter Protection. The network perimeter protection idea comes from the ancient technique of using walls such as Hadrian's Wall or the Great Wall for protecting a city. In networking parlance, a zone called a demilitarised zone (DMZ), aka a perimeter network, is created. All external untrusted users are restricted from using the services available in this zone. Typically, an organisation's public web server and authoritative DNS would reside in the DMZ. The rest of the network is partitioned into several security zones by a security architect. For example, a payment database would be deployed to an isolated network. Each zone is managed by one or more of the IDS, IPS or AG systems based on the significance of the information/infrastructure to be protected. Although without any tight control of the endpoints on the network, this has proven to achieve less separation than expected.

9 Advanced Network Security Topics

9.1 Software Defined Network, Virtualisation

Software Designed Networking (SDN) has become commonplace in data centres and other contexts for managing and controlling the network operation. In conventional IP network, routers perform both routing and forwarding functions. However, the SDN separates the packet forwarding functionality of the forwarding devices, i.e. the data plane from the control plane. The routing function and other intelligence is implemented in a centralised controller. On receiving of a new packet, the SDN switch requests for a forwarding rule from the controller. The switch then forwards all subsequent packets from the flow using this rule. The SDN architecture provides many new features to improve security for threat detection and attack prevention and provides innovative security services [17, 18].

For example, a DDoS attack can be inferred by the central controller more accurately, and a threat mitigation application may dynamically reprogram switches at the network perimeter to drop malicious traffic flows. A user on an infected machine can automatically be routed to a web-server issuing a quarantine notification. Another group of researchers has focussed on securing the SDN platform itself. The SDN controllers use a spanning tree algorithm (SPTA) for topology updates. In a DoS attack, an adversary could advertise a fake link and force the SPTA to block legitimate ports. Hong et al. [19] provide a number of attack vectors on practical SDN switch implementations.

SDN switches are prone to *a timing side channel attack*. Liu et al. [20] present several attack vectors. An attacker can send a packet and measure the time it takes the switch to process this packet. As discussed above, for a new packet, the switch will need to fetch a new rule from the controller, thus resulting in additional delay over the flows that already have rules installed at the switch. As an example, the attacker can determine whether an exchange between an IDS and a database server has taken place, or whether a host has visited a particular website. A possible countermeasure would introduce delay for thefirst few packets of every flow even if a rule exists [21]. SDN switches store rules in the cache memory for fast lookups. The rules are typically purged from the memory after a specified timeout period or removed due to certain other policy decisions. Liu et al. [20] also describe potential attacks by observing the cache rule removal behaviour. They suggest countermeasures such as a proactive rule setup or transforming the rule structure (e.g., merger) to make any inference difficult. Zerkane et al. [22] have methodically analysed and reported 114 SDN vulnerabilities.

A recent trend in networking is the use of Network Functions Virtualisation (NFV). The goal is to reduce capex and allow for the rapid introduction of new services to the market. Specialised network middleboxes such as firewalls, encoders/decoders, DMZs and deep packet inspection units are typically closed black box devices running proprietary software [23]. NFV researchers have proposed the deployment of these middleboxes entirely as virtualised software modules and managed via standardised and open APIs. These modules are called Virtual Network Functions (VNFs). A large number of possible attacks concern the Virtual Machine (Hypervisor) as well as configuring virtual functions. Lal et al. [24] provide a table of NFV security issues and best practice for addressing them. For example, an attacker can compromise a VNF and spawn other new VNFs to change the configuration of a network by blocking certain legitimate ports. Authors suggest hypervisor introspection and security zoning as mitigation techniques. Yang et al. [25] provide a comprehensive survey on security issues in NFV.

9.2 Internet of Things (IoT) Security

As discussed earlier, the Mirai malware shows how IoT devices such as IP cameras can be used to launch serious DDoS attacks. As it is an application driven field, vendors prefer 'first to market' with the resulting security being low priority. The other reason is that IoT devices are typically low-end and have limited capability for participating in advanced security protocols, especially when they are resource-constrained through battery power etc. Transport Layer Security (TLS) and Datagram TLS (DTLS) are cornerstones of IoT security. Prominent IoT application layer protocols adopt either TLS or DTLS as their security protocol in combination with Public Key Cryptography (PKC) or a Pre-Shared-Key (PSK) suite. These IoT application frameworks fulfill standard security requirements similar to traditional Internet applications. Since TLS requires a TCP connection, DTLS is widely used for limited bandwidth and lower reliability, as it is connectionless and UDP-based. While DTLS is designed to be used in constrained devices with limited communication capability, the End-to-End (E2E) communication manner of the DTLS causes scalability issues in large-scale IoT applications, especially under low-bandwidth standards such as IEEE 802.15.4. Given the emerging characteristics of heterogeneity, energy and performance, scalability, mobility and management, it is obvious that the current PKC with an E2E infrastructure will almost certainly not scale to accommodate future IoT applications [26]. E2E communication causes unscalable communication overheads and delays in large-scale applications. Furthermore, constrained devices are not equipped with adequate performance/memory to process resource-intensive PKC cryptography suites, thus resulting in performance/security degradation.

CROSS-REFERENCE OF TOPICS VS REFERENCE MATERIAL

	Kurose:2017 [1]	Stallings:2016 [4]	Taha15 [17]	4738466 [15]
1 Internet ArchitectureInternet Architecture	X			
2 Network Protocols and VulnerabilityNetwork Protocols and Vulnerability	X	X		
3 Application-Layer SecurityApplication Layer Security	X	X		
4 Transport-Layer SecurityTransport Layer Security	X	X		
5 Network Layer SecurityNetwork Layer Security	X	X		
6 Link Layer SecurityLink Layer Security	X	X		
7 Wireless LAN SecurityWireless LAN Security	X	X		
8 Network Defence ToolsNetwork Defence Tools		X		X
9 Advanced Network Security TopicsAdvanced Network Security Topics			X	

REFERENCES

[1] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach (7th Edition)*, 7th ed. Pearson, 2017.

[2] D. Dolev and A. C. Yao, "On the security of public key protocols," vol. 29, no. 2, pp. 198–208, March 1983.

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher,

C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[4] W. Stallings, *Network Security Essentials, Applications and Standards (6th Edition)*, 6th ed. Pearson, 2016.

[5] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL landscape: A thorough analysis of the X.509 PKI using active and passive measurements," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 427–444. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068856

[6] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2046–2069, Fourth 2013.

[7] T. Chung, R. van Rijswijk-Deij, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Understanding the role of registrars in DNSSEC deployment," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: ACM, 2017, pp. 369–383. [Online]. Available: http://doi.acm.org/10.1145/3131365.3131373

[8] C. Kiraly, S. Teofili, G. Bianchi, R. Lo Cigno, M. Nardelli, and E. Delzeri, "Traffic flow confidentiality in IPsec: Protocol and implementation," in *The Future of Identity in the Information Society*, S. Fischer-Hübner, P. Duquenoy, A. Zuccato, and L. Martucci, Eds. Boston, MA: Springer US, 2008, pp. 311–324.

[9] S. Kent, C. Lynn, and K. Seo, "Design and analysis of the secure border gateway protocol (s-bgp," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*.

[10] C. Hall, R. Anderson, R. Clayton, E. Ouzounis, and P. Trimintzios, "Resilience of the internet interconnection ecosystem," in *Economics of Information Security and Privacy III*. Springer, 2013, pp. 119–148.

[11] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[12] T. Kiravuo, M. Sarela, and J. Manner, "A survey of Ethernet LAN security," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1477–1491, Third 2013.

[13] A. Stubblefield, J. Ioannidis, and A. D. Rubin, "Using the Fluhrer, Mantin, and Shamir attack to break WEP," in *NDSS*, 2002.

[14] E. Tews and M. Beck, "Practical attacks against WEP and WPA," in *Proceedings of the Second ACM Conference on Wireless Network Security*, ser. WiSec '09. New York, NY, USA: ACM, 2009, pp. 79–86. [Online]. Available: http://doi.acm.org/10.1145/1514274.1514286

[15] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, Fourth 2008.

[16] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE Symposium on Security and Privacy*, May 2010, pp. 305–316.

[17] S. Taha Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Transactions on Reliability*, vol. 64, pp. 1–12, 09 2015.

[18] A. Shaghaghi, M. A. Kaafar, and S. Jha, "Wedgetail: An intrusion prevention system for the data plane of software defined networks," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: ACM, 2017, pp. 849–861. [Online]. Available: http://doi.acm.org/10.1145/3052973.3053039

[19] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures," in *Proceedings 2015 Network and Distributed System Security Symposium*. Internet Society, 2015. [Online]. Available: https://doi.org/10.14722%2Fndss.2015.23283

[20] S. Liu, M. K. Reiter, and V. Sekar, “Flow reconnaissance via timing attacks on SDN switches,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 196–206.

[21] H. Cui, G. O. Karame, F. Klaedtke, and R. Bifulco, “On the fingerprinting of software-defined networks,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2160–2173, 2016.

[22] S. Zerkane, D. Espes, P. Le Parc, and F. Cuppens, “Vulnerability analysis of software defined networking,” in *Foundations and Practice of Security*, F. Cuppens, L. Wang, N. Cuppens-Boulahia, N. Tawbi, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2017, pp. 97–116.

[23] Z. G. A. Gember, P. Prabhu and A. Akella, “Toward software defined middlebox networking,” in *In Proceedings of the 11th ACM Workshop on Hot Topics in Networks (HotNets)*, Redmond, WA, 10 2012, pp. 7–12.

[24] S. Lal, T. Taleb, and A. Dutta, “NFV: Security threats and best practices,” *IEEE Communications Magazine*, vol. 55, no. 8, pp. 211–217, Aug 2017.

[25] W. Yang and C. Fung, “A survey on security in network functions virtualization,” in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 15–19.

[26] J. Y. Kim, W. Hu, D. Sarkar, and S. Jha, “ESIoT: Enabling secure management of the internet of things,” in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec ’17. New York, NY, USA: ACM, 2017, pp. 219–229. [Online]. Available: http://doi.acm.org/10.1145/3098243.3098252