**Week 7 - Chapter 7 - L\_PKI\_TLS\_Introduction** **[ Public Key Infrastructure ]**

**A person in a suit

Description automatically generatedA diagram of a hacker

Description automatically generated**

**2 ->** Giving Alice’s public key to Bob allows Bob to send secret data to Alice, because Bob can encrypt it using Alice’s public-key. However, is this really secure?

**3 ->** **Problem**: how can Bob be sure that the public-key is coming from Alice? The Internet can be sniffed and changed by attacker easily. This is called the man-in-the-middle attack. Attacker Mallory can pretend to be Bob when talking to Alice and pretend to be Alice when talking to Bob.

**A white paper with black text

Description automatically generatedA close up of a document

Description automatically generated**

**4 ->** The fundamental issue here is that Bob has no way to verify whether the received public-key belongs to Alice or not.

If some well-known person can generate a proof that this public-key belongs to Alice, then Bob will be convinced. This is the idea of public-key certificate.

**5 ->** Return to the man-in-the-middle attack problem between Alice and Bob. The above procedure shows how this attack is no longer working, if Alice sends her public-key certificate to Bob.

**A close up of a text

Description automatically generatedA close up of a certificate

Description automatically generated**

**6 ->** Certificate Authority (CA) is the entity providing the certificate service.

The standard certiticate format is to use x.509. In our certificate experiment, you will practice how this can be done.

**7 ->** To verify a certificate, we need the issuer (i.e., CA) public-key. This CA’s public-key is presented in its own certificate.

**8 ->** In our experiment, you will play as a root CA to issue certificates.

The best-known root certificates are distributed in operating systems by their manufacturers. [Microsoft](https://en.wikipedia.org/wiki/Microsoft) distributes root certificates belonging to members of the Microsoft Root Certificate Program to [Windows](https://en.wikipedia.org/wiki/Windows) desktops and [Windows Phone 8](https://en.wikipedia.org/wiki/Windows_8_phone). Apple distributes root certificates belonging to members of its own [root program](https://en.wikipedia.org/wiki/Public_key_certificate).

**9 ->** TLS evolved from the older version SSL and is gradually replacing SSL

When SSL was standardized by IETF, it was renamed to TLS

SSL version 3.0 was deprecated and replaced by TLS in June 2015

TLS is designed to run on top of TCP but can be implemented with other protocols such as UDP

**A certificate with text and words

Description automatically generated with medium confidenceA certificate with text on it

Description automatically generated**

**Transport Layer Security**

**A close-up of a document

Description automatically generatedA screenshot of a document

Description automatically generated**

**12 ->** Application will use TLS\_socket.send(msg) to send message.

* TLS will take msg from application and do encryption and integrity operations.
* After this, a ciphertext of msg is computed. TLS then sends using TCP\_socket.send(ciphertext)
* At receiver side, TLS will verifies the integrity of ciphertext and decrypt it to msg. Finally, it gives to application which will be received via TLS\_socket.recv().

**13 ->** In the previous slide, we did not mention how the encryption key is shared between client and server, what algorithm for encryption and integrity will be used

* All these will be done in the TLS Handshake protocol. Imagine that TCP connection will run 3-way handshake protocol to set up the connection as well as its parameters such as receive window. TLS handshake protocol has the similar purpose.

**A white paper with black text

Description automatically generatedA diagram of a software process

Description automatically generated with medium confidence**

**14 -> Ciphert suites:** the possible encryption algorithms (such as RSA, AES, MD5, SHA2). Client-server will need to agree on what algorithms will be used. Client hello asks this to server and server makes decision on server hello.

* **Client random:** a random number chosen by client. This random will always be put into the encryption and integrity operation so that the ciphertext is different from old one (so attacker can not use older ciphertext to cheat the receiver).
* **Server certificate:**  server will provide his public-key certificate to client. This will be verified as mentioned before.
* **Client key exchange:** this will be key exchange between client and server so that they will share a secret key.

**A screenshot of a computer

Description automatically generatedA close-up of a document

Description automatically generated**

**15 ->** TLS runs on top of TCP. So before TLS, TCP connection should be established.

**16 ->** When client receives the certificate, it will verify it as mentioned before.

* In addition, client will verify that the hostname it wants to talk with is the same as that carried in the certificate.

**A screenshot of a computer

Description automatically generatedA diagram of a computer program

Description automatically generated**

**17** -> In this section, we focus on the application record type, which is used to transfer application data between a client and a server

**Content Type:** Indicates the type of protocol data carried by current record (handshake, alert, application, heartbeat or ChangeCipherSpec)

**Version:** This field identifies major and minor version of TLS being used

**Length:** The length of the payload field.

|  |  |  |
| --- | --- | --- |
| **A screen shot of a computer program  Description automatically generated** | 1. **-> cadir:** it is the location you store CA certificates (recall CA’s public-key will be used to verify the server public-key certificate).  * **ssl.CERT\_REQUIRED:** need to verify server certificate * **check\_hostname:** need to check sys.argv[1] is the same as the owner carried in server certificate. | |
| **A screen shot of a computer program  Description automatically generated** | |  |

**Week 8 - Chapter 8 - L\_Firewalls-1**   **[ Firewalls ]**

A white background with black text

Description automatically generatedA black text on a white background

Description automatically generated

A white background with black text

Description automatically generated

A white text on a white background

Description automatically generated

A black text on a white background

Description automatically generated

A black text on a white background

Description automatically generated

A computer screen shot of a computer network

Description automatically generated

A computer screen shot of a computer

Description automatically generated

|  |  |
| --- | --- |
| A diagram of a firewall  Description automatically generated | **Limitation :** Need to implement new proxies to handle new protocols. Slower compared to other firewalls  **Advantage :** Ability to authenticate users directly rather than depending on network addresses of the system. Reduces the risk of IP spoofing attacks that are easy to launch against a network. |

A white background with black text

Description automatically generated

A diagram of a network

Description automatically generated

* **NF\_IP\_LOCAL\_OUT:** The packets generated by the current host start its way out to Internet from here.
* **NF\_IP\_POST\_ROUTING:** the packets are about to be out of the host and entering a different network.
* **NF\_IP\_PRE\_ROUTING:** All packets will go through this hook
* **NF\_IP\_LOCAL\_IN:** This hook is called when a packet is destined to the machine itself
* **NF\_IP\_FORWARD:** This hook is called if the packet is destined to other hosts (not the current host).
* **Routing:** This is routing processing, which either lets in the current host or lets it continue the traverse.
* **Network Stack:** This is where the packet goes through the network layer, transport layer and application layer. Here it **ends** its traverse at application program socket.

|  |  |
| --- | --- |
| A white and black text with black text  Description automatically generated with medium confidence | **PREROUTING:** for altering incoming packets before routing;  **OUTPUT:** for altering locally-generated packets before routing;  **INPUT:** for packets coming into the box itself,  **FORWARD:** for altering packets being routed through the box;  **POSTROUTING:** for altering packets as they are about to go out. |

A close up of a text

Description automatically generated

A diagram of a computer program

Description automatically generated

In the figure, option 1, 2, 3, 4 are explained as follows.

1. Do the routing decision: the current packet is going to the current host or going to other hosts.
2. Decide which interface to send out the packet.
3. Take the packet in as destined to the current host
4. Let the packet continue traverse to other hosts.

Since filter, NAT and mangles are independents firewall, they all go through Netfilter basic structure. It turns out, a Netfilter hook could be processed by more than one times.

For example, NF\_IP\_PRE\_ROUTING hook is implemented by mangle and nat both. However, the processings by NAT and MANGLE are certainly different.

|  |  |
| --- | --- |
| A black text on a white background  Description automatically generated  A screenshot of a computer  Description automatically generated | **17 ->** -j: it executes the **target** rule TTL –ttl-inc 5, which will increase TTL field of a packet by 5.  In general, this will have format:  -j **target** *target-opt*, where **target** is the program (or module) name and *tart-opt* is its option.  The possible **target** modules are described in man iptables-extensions. You can see in the description that TTL module is only available for **mangle** table.  Also, **target** can be ACCEPT, DROP or RETURN. Here ACCEPT means to let the packet through. DROP means to drop the packet on the floor. RETURN means stop traversing this chain and resume at the next rule in the previous (calling) chain. |

A black text on a white background

Description automatically generated

A screenshot of a computer program

Description automatically generated

|  |  |
| --- | --- |
| A screenshot of a computer error  Description automatically generated  A screenshot of a computer program  Description automatically generated | **20 ->** -P define the default policy. The general format is  -P **chain\_name** **ACCEPT**\_or\_**DROP**  -t TABLE\_NAME is ignored in the above slides. Then, it applies to the default table **filter**. If we want to apply this default policy to other tables, you need to specify the table name: e.g., -t nat  **21 ->** -p ***protocol*** *option. Ex.* –p tcp --sport 23*; see iptables-extensions for description.*  *The above rule can be read as*  Append a rule to **input** chain (of default table filter). This rule is: for protocol **tcp** with dport 22, apply the target decision **accept**. |

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**24 ->** After the experiment, remove all the rules by allowing traffic on all the chains and flushing out the existing configurations

**27->** The earlier firewall allowed all the outgoing TCP traffic. An attacker who compromises an internal host can exfiltrate data over TCP. The connection cannot be made as incoming traffic is blocked. But this is sufficient for exfiltrating data. To avoid such attacks, we need to setup stateful firewalls

A screenshot of a computer program

Description automatically generated

A close-up of a white background

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

**Week 9 - Chapter 9 – Web\_CSRF** **[ Cross Site Request Forgery (CSRF) ]**A close-up of a website

Description automatically generatedA screenshot of a webpage

Description automatically generated

A close up of a web page

Description automatically generatedA close-up of a white paper

Description automatically generated

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

A close-up of a computer screen

Description automatically generatedA screenshot of a computer error

Description automatically generated

A white background with black text

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generatedA white paper with black text

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

**15 ->**

* We create a new <form> element using p=document.createElement( 'form'). The properties are explained as follows.
* document.body.appendChild(**p**): This will add the form **p** to the page.
* **p.action**: this sets the URL where the form will be submitted to
* **p.method**: this sets the HTTP request method when sending to the server. Here we use the post method.
* **p.innerHTML**: this allows you to define or modify the HTML content inside the element p. This is the HTML content for the form.
* E.g., <div id="my-div"> <hl>New Heading</hl> <p>New paragraph content.</p> </div>
* Then, **p.innerHTML**=“<hl>New Heading</hl> <p>New paragraph content.</p> “
* **p.submit()**: Then, we call p.submit() to submit the form using JavaScript. This is equivalent to clicking the submit button in the form.
* **window.onload**: window.onload is an event that fires when the entire webpage (including all of its resources, such as images and scripts) has finished loading. It is a global event that is triggered on the window object, which represents the browser window. In our example, the event is the function forge\_post().

A screenshot of a profile

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generatedA white background with black text

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generatedA white paper with black text

Description automatically generated

A close-up of a computer screen

Description automatically generatedA white text on a white background

Description automatically generated

A screenshot of a computer program

Description automatically generatedA close-up of a computer code

Description automatically generated

**Week 10 - Chapter 10 - L\_BitcoinBlockchain** **[ Bitcoin and Blcokchain]**

A white background with black text

Description automatically generatedA screen shot of a computer

Description automatically generated

A diagram of a blockchain

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a chat

Description automatically generatedA diagram of a blockchain

Description automatically generated

A white text with black text

Description automatically generatedA screenshot of a computer code

Description automatically generated’

11 -> Note to instructor: using a board or drawing directly on the slide to show how the script execute

21 -> Block\_Hash is **Hash256**(Hash256(**header**)), similar to txID.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a program

Description automatically generatedA screenshot of a computer code

Description automatically generated

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer program

Description automatically generated

A white background with black text

Description automatically generatedA diagram of blockchain

Description automatically generated

A diagram of a blockchain

Description automatically generatedA diagram of a blockchain

Description automatically generated

A white background with black text

Description automatically generatedA diagram of blockchain

Description automatically generated

A diagram of a tree

Description automatically generatedA diagram of a blockchain

Description automatically generated

A diagram of a block chain

Description automatically generatedA diagram of a block chain

Description automatically generated

A table with numbers and text

Description automatically generated with medium confidenceA diagram of blockchain

Description automatically generated

A white background with black text

Description automatically generated

**Week 11 - Chapter 11 – Smart\_Contract**  **[ Smart Contract]**

**A close-up of a text

Description automatically generatedA close up of a sign

Description automatically generated**

**A close-up of a contract

Description automatically generatedA white paper with black text

Description automatically generated**

**A screenshot of a computer screen

Description automatically generatedA white background with black text

Description automatically generatedA white background with black text

Description automatically generatedA screenshot of a computer

Description automatically generated**

***3 ->*** H is **Keccak-256** hash function. For both addresses computation, it computes Keccak-256 hash and then outputs the last 20 bytes (i.e., 160-bit). The CreattorNonce is the number of transactions the CreatorAddr has done.

***9 ->*** Another popular choice of development platform is **Truffle** (on your local computer) using the blockchain simulated by **Ganache**. Truffle provides a more comprehensive and feature-rich development environment suitable for larger projects, while Remix offers a lightweight and accessible web-based IDE for quick prototyping and experimentation.

Our teaching uses Remix to get away from the complication from things other than the smart contract itself.

***10 ->*** **Pragma** **solidity** is compiler directive that tell you which version of solidity can be used to compile the program. In the above program, it indicates that at least solidity 0.8.2 but should not use solidity 0.9.0 or higher version. You might also see **solidity ^0.8.2** in a program, this asks to use the **exact** solidity 0.8.2 for the compilation. The file name better uses the contract name to be more informative.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer code

Description automatically generated**A screen shot of a computer

Description automatically generated**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA text on a white background

Description automatically generated’A text on a white background

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA white paper with black text

Description automatically generated

A screenshot of a computer screen

Description automatically generatedA white background with black text

Description automatically generated

A screenshot of a computer

Description automatically generatedA white text box with black text

Description automatically generated

A screenshot of a message

Description automatically generatedA screenshot of a computer code

Description automatically generated

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

***11 ->*** **uint256** is the same as **uint** which can take the maximum value 2^256-1. **int** (which is **int256**) takes value between -2^255 to 2^255-1. You might see **constant** function modifier. Somewhere. This is the older version of **view** type and deprecated now. **view** type function can be called without any cost (no gas usage and no signature) and you can call it by your local node (if it stores the whole blockchain) without broadcasting to the network.

***12 ->*** Public function can be executed by any account (either an **EOA** or **CA**). In contrast, if you change **public** to **private** for retrieve( ), then after deploy, it is the function is no longer accessible. You might also see **external** functions. External function can only be called externally while public function can be called by external and internal functions. You might also see **internal** function. Internal function can be called by current contract or its derived contract while private function can not be called by derived contract. Variable **number** (called storage variable, part of the contract state) can also be accessed by a *getter* function **number( ).**

***13 ->*** When you are familiar with solidity, you can use Javascript tool Truffle to automatically deploy the smart contract.

***28 ->*** You also have constructor with input: constructor (uint8 \_number) public { number=\_number; } . You will be asked to provide the input **\_number** when you deploy the contract.

***30 ->*** The event usually is used to monitor the contract state change. For example, with event **tx\_sender** event, you know **owner**’s balance has been changed.

***31 ->*** nonce must match current nonce of sender in sender’s account data. When Tx processed successfully, nonce in sender’s account data is incremented by 1.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

A white text with black text

Description automatically generatedA white background with black text

Description automatically generated

A diagram of a blockchain

Description automatically generatedA white background with black text

Description automatically generated

A white background with black text

Description automatically generated’