# CYBER SECURITY & CRYPTOGRAPHY (part-3)

techworldthink • March 07, 2022

## 7. Describe security association of IPSec.

Internet Protocol Security (IPsec) is a secure network protocol suite that authenticates and encrypts the packets of data to provide secure encrypted communication between two computers over an Internet Protocol network. It is used in virtual private networks (VPNs).

Security Association(SA)- It is a temporary communication link b/w two systems. It is a one way communication of client and server.

Client -> <- Server . Here there are 2 Security Associations are there in b/w client & server

*Security Association parameters*

To identify Security Association 3 , parameters are there.

1.Security Parameter Index(SPI).This will be carried out in both AH & ESP protocol. SPI is the unique number given to SA.

2. IP Destination address. Address of the destination.

3 . Protocol Identifier: This specifies which protocol the SA is used. i.e ESP/AH.

Parameters which are associated with SA are stored in Security Association Database(SAD)

1.SPI- unique identification of particular SA

2.Sequence Number Counter.

3.Sequence Number Overflow- where to stop the message.

4. Anti-replay Window- to avoid receiving the duplicate packets

5. AH information- It gives information about which are the authentication & signatures algorithms used in the AH protocol

6.ESP information

7. Life time of SA

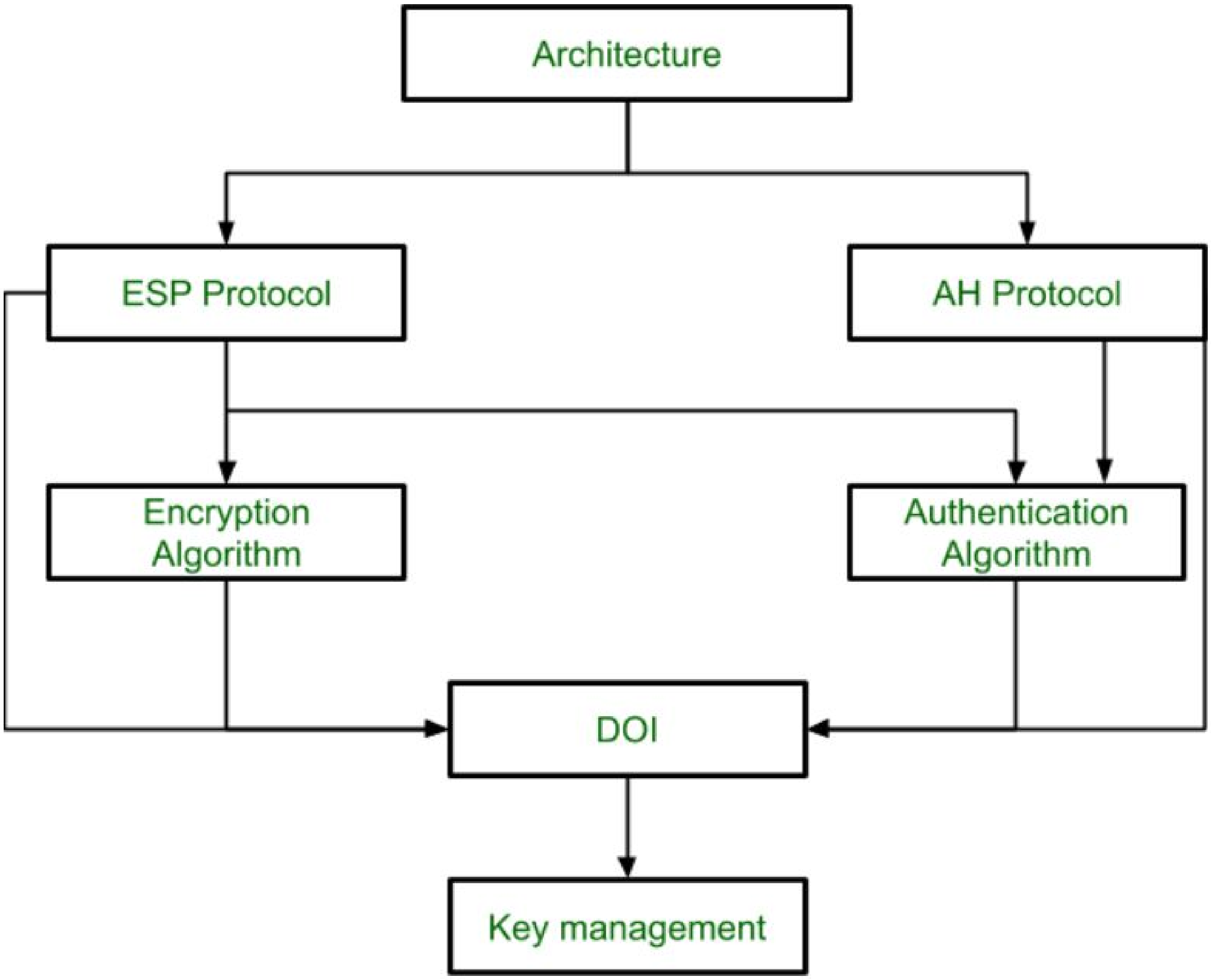
8. IPSec protocol mode- Transportmode, Tunnel mode

**IPSec Architecture**

IPSec (IP Security) architecture uses two protocols to secure the traffic or data flow. These protocols are ESP (Encapsulation Security Payload) and AH (Authentication Header). IPSec Architecture include protocols, algorithms, DOI, and Key Management. All these components are very important in order to provide the three main services:

- Confidentiality
- Authentication
- Integrity

**IP Security Architecture:**



**1. Architecture:**

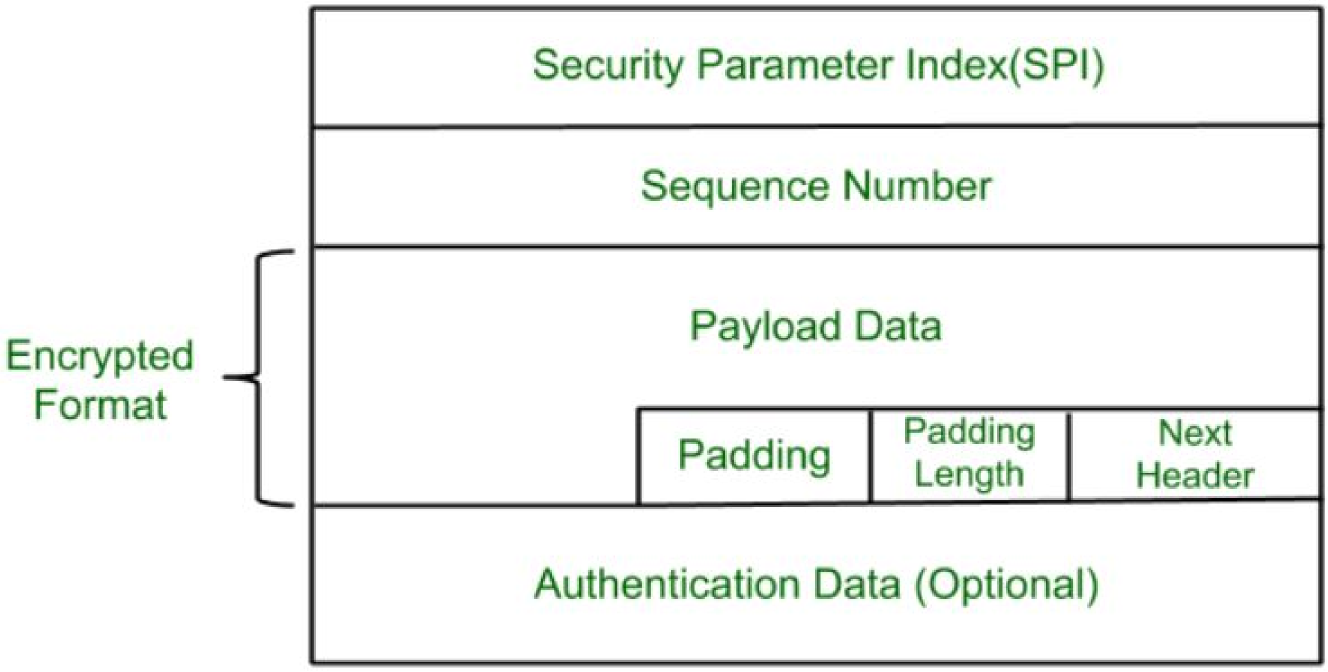
Architecture or IP Security Architecture covers the general concepts, definitions, protocols, algorithms and security requirements of IP Security technology.

**2. ESP Protocol:**

ESP(Encapsulation Security Payload) provide the confidentiality service. Encapsulation Security Payload is implemented in either two ways:

- ESP with optional Authentication.
- ESP with Authentication.

**Packet Format:**



- **Security Parameter Index(SPI):**

This parameter is used in Security Association. It is used to give a unique number to the connection build between Client and Server.

- **Sequence Number:**

Unique Sequence number are allotted to every packet so that at the receiver side packets can be arranged properly.

- **Payload Data:**

Payload data means the actual data or the actual message. The Payload data is in encrypted format to achieve confidentiality.

- **Padding:**

Extra bits or space added to the original message in order to ensure confidentiality. Padding length is the size of the added bits or space in the original message.

- **Next Header:**

Next header means the next payload or next actual data.

- **Authentication Data**

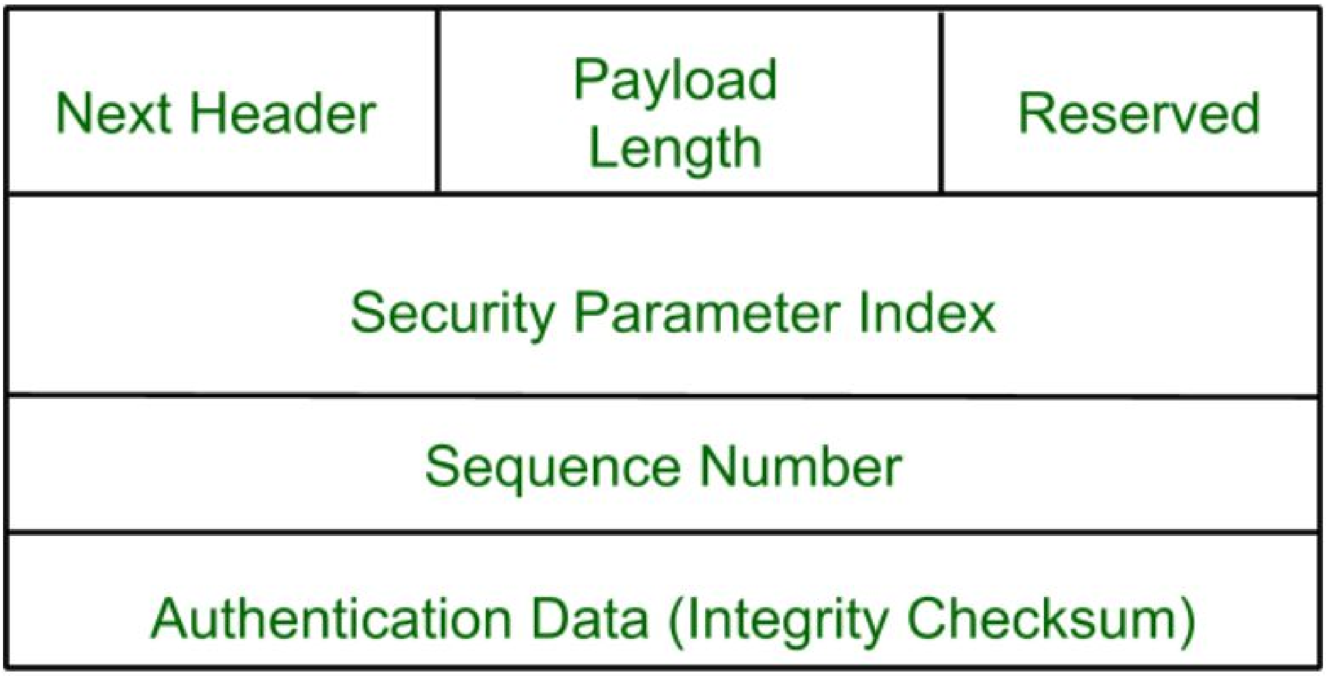
This field is optional in ESP protocol packet format.

**3. Encryption algorithm:**

Encryption algorithm is the document that describes various encryption algorithm used for Encapsulation Security Payload.

**4. AH Protocol:**

AH (Authentication Header) Protocol provides both Authentication and Integrity service. Authentication Header is implemented in one way only: Authentication along with Integrity.



<table>
<tr><td>Next Header</td><td>Payload Length</td><td>Reserved</td></tr>
<tr><td colspan="3">Security Parameter Index</td></tr>
<tr><td colspan="3">Sequence Number</td></tr>
<tr><td colspan="3">Authentication Data (Integrity Checksum)</td></tr>
</table>

Authentication Header covers the packet format and general issue related to the use of AH for packet authentication and integrity.

**5. Authentication Algorithm:**

Authentication Algorithm contains the set of the documents that describe authentication algorithm used for AH and for the authentication option of ESP.

**6. DOI (Domain of Interpretation):**

DOI is the identifier which support both AH and ESP protocols. It contains values needed for documentation related to each other.

**7. Key Management:**

Key Management contains the document that describes how the keys are exchanged between sender and receiver.

## 8. Explain about S/MIME.

***Secure/Multipurpose Internet Mail Extensions (S/MIME)***

MIME stands for Multipurpose Internet Mail Extensions. It is used to extend the capabilities of Internet e-mail protocols such as SMTP. The MIME protocol allows the users to exchange various types of digital content such as pictures, audio, video, and various types of documents and files in the e-mail. MIME was created in 1991 by a computer scientist named Nathan Borenstein at a company called Bell Communications.

MIME is an e-mail extension protocol, i.e., it does not operate independently, but it helps to extend the capabilities of e-mail in collaboration with other protocols such as SMTP. Since MIME was able to transfer only text written file in a limited size English language with the help of the internet. At present, it is used by almost all e-mail related service companies such as Gmail, Yahoo-mail, Hotmail.

1. The MIME protocol supports multiple languages in e-mail, such as Hindi, French, Japanese, Chinese, etc.
2. Images, audio, and video cannot be sent using simple e-mail protocols such as SMTP. These require MIME protocol.

S/MIME is a protocol for the secure exchange of e-mail and attached documents originally developed by RSA Security. Secure/Multipurpose Internet Mail Extensions (S/MIME) adds security to Internet e-mail based on the Simple Mail Transfer Protocol (SMTP) method and adds support for digital signatures and encryption to SMTP mail to support authentication of the sender and privacy of the communication. Note that because HTTP messages can transport MIME data, they can also use S/MIME.

S/MIME is an extension of the widely implemented Multipurpose Internet Mail Extensions (MIME) encoding standard, which defines how the body portion of an SMTP message is structured and formatted. S/MIME uses the RSA public key cryptography algorithm along with the Data Encryption Standard (DES) or Rivest-Shamir-Adleman (RSA) encryption algorithm.

## 9. How can we prevent Injection attack?

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application.

A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands.

Here is a basic HTML login form with two inputs: `username` and `password`.

```
<form method="post" action="/login">
<input name="username" type="text" id="username">
<input name="password" type="password" id="password">
</form>
```

The common way for the `/login` to work is by building a database query. If the variables `$request.username` and `$request.password` are requested directly from the user’s input, this can be compromised.

```
SELECT id
FROM Users
WHERE username = '$request.username'
AND password = '$request.password'
```

For example, if a user inserts `admin' or 1=1 --` as the username, he/she will bypass the login form without providing a valid username/password combination.

```
SELECT id
FROM Users
WHERE username = 'admin' or 1=1--
AND password = 'request.password'
```

The issue is that the `'` in the `username` closes out the `username` field, then the `--` starts a SQL comment causing the database server to ignore the rest of the string. As the inputs of the web application are not sanitized, the query has been modified in a malicious way.

**How to Prevent SQL Injection**

The source of the problem of SQL Injection (the most important injection risk) is based on SQL queries that use untrusted data without the use of parametrized queries (without `PreparedStatement` in Java environments).

First of all Hdiv minimizes the existence of untrusted data thanks to the web information flow control system that avoids the manipulation of the data generated on the server side. This architecture minimizes the risk to just the new data generated legally from editable form elements. It's important to note that even using `PreparedStatement` if the query is based on untrusted data generated previously at server side (for instance the identification id of an item within a list) it's possible to exist a SQL Injection risk.

Although `PreparedStatement` solves the most of the cases, there are some SQL keywords that can not be used with `PreparedStatement`, such as `ORDER BY`. In these cases, you have to concatenate the column name and the order to the SQL query but only after verifying that the column name and order are valid in this context and sanitising them to counter any attempt of SQL Injection attack.

## 10. What is XXE? How to prevent it?

XML external entity injection (also known as XXE) is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data. It often allows an attacker to view files on the application server filesystem, and to interact with any back-end or external systems that the application itself can access.

In some situations, an attacker can escalate an XXE attack to compromise the underlying server or other back-end infrastructure, by leveraging the XXE vulnerability to perform server-side request forgery (SSRF) attacks.

Some applications use the XML format to transmit data between the browser and the server. Applications that do this virtually always use a standard library or platform API to process the XML data on the server. XXE vulnerabilities arise because the XML specification contains various potentially dangerous features, and standard parsers support these features even if they are not normally used by the application.

There are various types of XXE attacks:

- Exploiting XXE to retrieve files, where an external entity is defined containing the contents of a file, and returned in the application's response.
- Exploiting XXE to perform SSRF attacks, where an external entity is defined based on a URL to a back-end system.
- Exploiting blind XXE exfiltrate data out-of-band, where sensitive data is transmitted from the application server to a system that the attacker controls.
- Exploiting blind XXE to retrieve data via error messages, where the attacker can trigger a parsing error message containing sensitive data.

**How to prevent XXE vulnerabilities**

Virtually all XXE vulnerabilities arise because the application's XML parsing library supports potentially dangerous XML features that the application does not need or intend to use. The easiest and most effective way to prevent XXE attacks is to disable those features.

Generally, it is sufficient to disable resolution of external entities and disable support for `XInclude`. This can usually be done via configuration options or by programmatically overriding default behavior. Consult the documentation for your XML parsing library or API for details about how to disable unnecessary capabilities.