Jimma University

Jimma Institute of Technology

Faculty of Electrical and Computer Engineering

Masters of Science in Computer Engineering

Advanced computer and Network Security

Assignment one

Researches on network security Attack

Prepared by:

Misganaw Aguate & Fasika Tegegn

Submitted to: - Dr. Henock M. (PHD)

Abstract

It is necessary identifying and classifying different types of network attack type on both wired and wireless network for the purpose of detection and mitigation. It our world there is a lot of network security attack that becomes a series threat for safe communication between to or more than interconnected computing device through network. In this paper we are going to list down and classify network security attack based on three main categories and we select three type of attack specifically (***Wormhole attack, session hijacking and SQL injection***) then discuss on brief description of working principles, which implies how the attack type work and what intrusion is done by the. Finally we finalize the paper with the mitigation mechanism of selected attack type. Even though there are many mechanisms to detect and solve this type of attack, we select the recent mitigation technique to make our work short and precise in easy and understandable form.

Contents

Contents page

Lists of attack

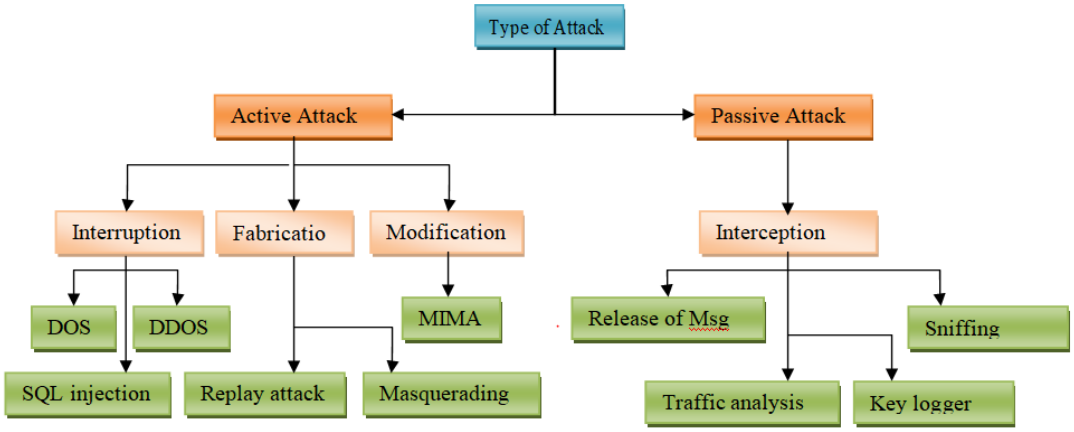
The following are some of the available network security threats among multiple one.

- Denial of service (DOS)
- Distributed denial of service (DDOS)
- SQL injection
- Replay attack
- Masquerading
- Release of message
- Jamming
- Packet sniffing
- Traffic analysis
- Key logger
- Eavesdropping
- Man in the middle attack (MITM)
- Disruption of MAC
- Attack on rout discovery protocol (ARDP)
- Resource consumption
- SYN flooding
- Session hijacking
- Attack on routing table overflow (ARTO)
- Attack on rout maintenance (ARM)
- Attack on data forwarding (ADF)
- Location disclosure
- Malicious software
- Malicious user
- Repudiation attack
- Black hole attack
- Gray hole attack
- Jellyfish attack
- Cooperative black hole attack
- Wormhole attack
- Sybil attack
- Rushing attack
- Hello flood
- Cache poisoning
- Bogus registration
- Byzantine attack
- Social engineering

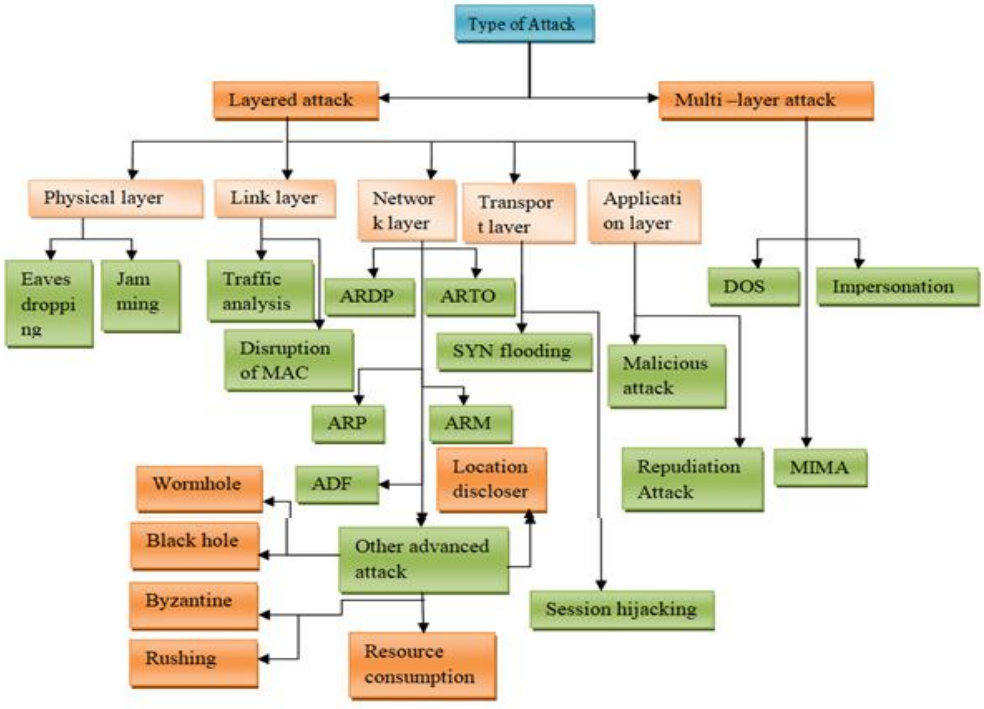
Classifications of attack

- Basic classification
- Classification based on OSI layer
- Classification based on MANET (network layer)

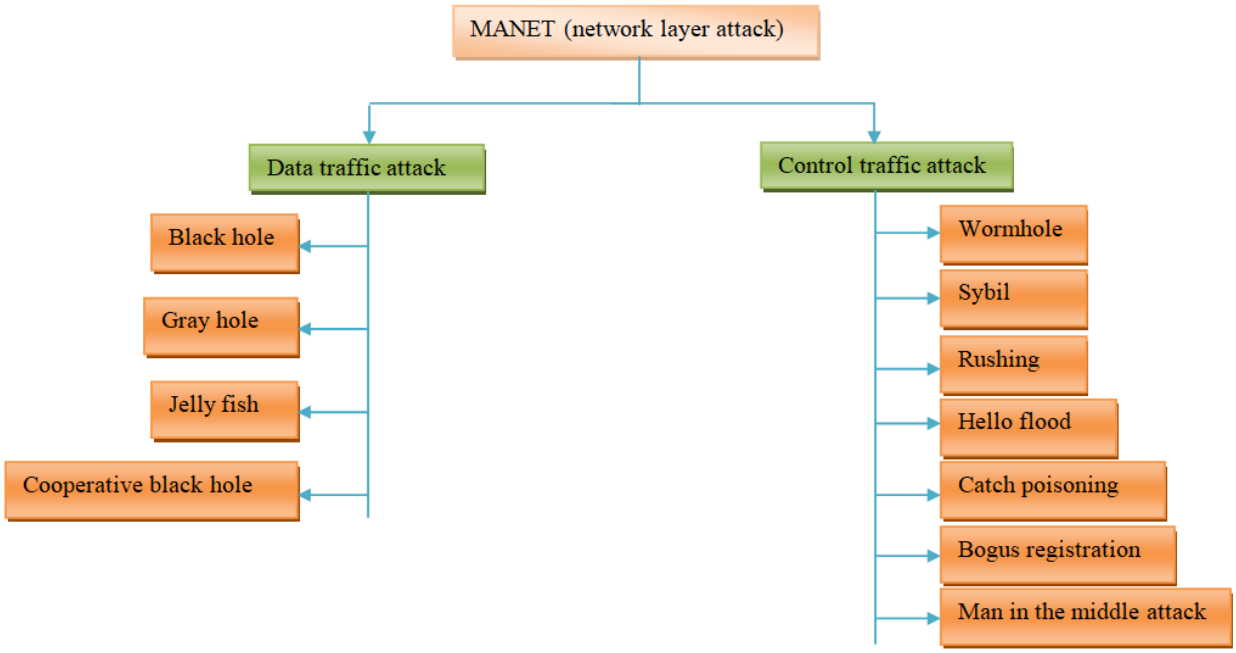
Basic classification



Classification based on OSI layer



Classification based on MANET (Network layer)



All the above classification of attack are what we have read and identify some of available network security attacks now a day. Among this a lot of attack we are going to select ***wormhole attack, session hijacking and SQL injection*** for the following research format.

Name of attack:	
Types of attack:	
Dates of attack:	
Computers/organizations affected:	
How it works and what it did:	
Mitigations options:	
References and info links:	

1. Wormhole attack

Name of attack: Wormhole attack

Type of attack:

Type Basic classification	Type based on OSI layer	Type based on MANET(network layer)
Active attack (DOS)	Network layer attack	Network layer attack

Date of attack: February 2006 G.C

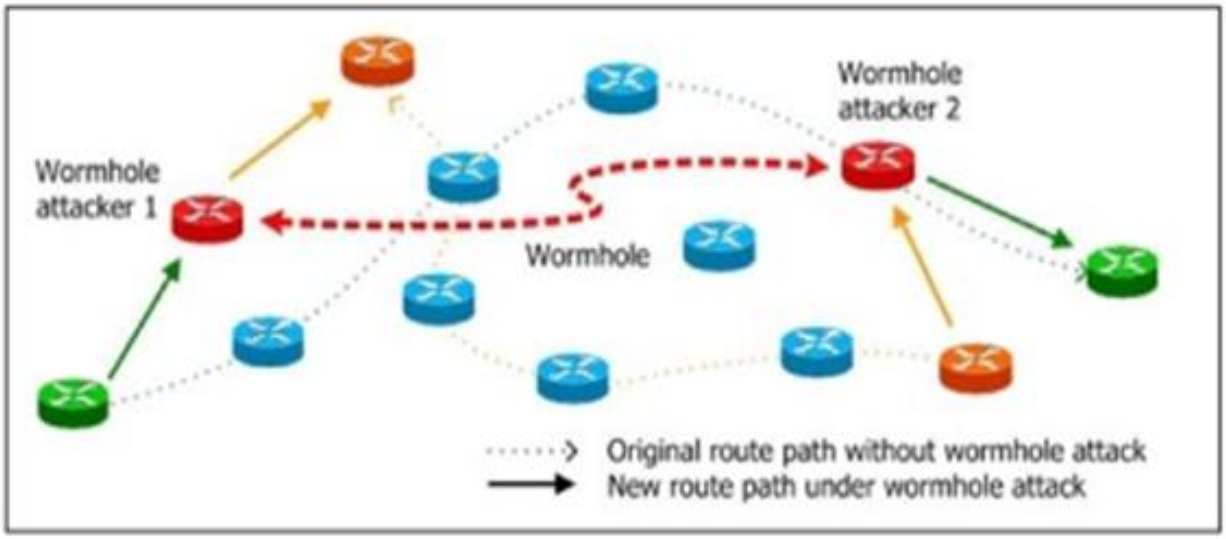
Affected organization/computer: Wireless networks, especially against many ad hoc network routing protocols and location-based wireless security systems. This includes any computing device which interconnect on wired or wireless network

1.1. Working principle

Worm hole, in cosmological term, connects two distant points in space via a shortcut route. In the same way in MANET also one or more attacking node can disrupt routing by short circuiting the network, thereby disrupting usual flow of packets. If this link becomes the lowest cost path to the destination then these malicious nodes will always be chosen while sending packets to that destination. The attacking node then can either monitor the traffic or can even disrupt the flow (via one of the DATA traffic attack). Wormhole attack can be done with single node also but generally two or more malicious node connects via a *wormhole-link* [1].

In a wormhole attack, attackers "tunnel" packets to another area of the network bypassing normal routes as shown in the following figure[2]. In practice, attackers can use high power antennas or a wired link, or other methods. The resulting route through the wormhole may have a better metric, i.e., a lower hop-count than normal routes. With this leverage, attackers using wormholes can easily manipulate the routing priority in MANET to perform the following attack [2].

- Eavesdropping
- Packet modification
- Perform a DoS (Denial of Service) attack
- The entire routing system in MANET can even be brought down using the wormhole attack.



1.2. Mitigation option

Even though there is a lot of published solution to defend against wormhole attack, for our discussion we list out the following sample:

- Geographical leash and temporal leash
- Directional antenna
- Multi-Dimensional Scaling -Visualization of Wormhole
- Local Broadcast Key
- Graph theoretic approach
- Multipath Hope count analysis

Generally protecting against wormhole attack follows to basic methodology:

- Viewpoint of administrator and
- Viewpoint of users and utilize routing information already available in standards like RFC3561

Viewpoint of administrator

Trying to identify the wormhole, and then defend against it. They can further be classified as centralized systems like MDS-VOW (*Multi-Dimensional Scaling -Visualization of Wormhole*) [3] and distributed systems such as LBK (*Local Broadcast Key*) [4]. Some require substantial calculation and some others employ special nodes in the network. These methods consume effort and bandwidth as overhead.

Viewpoint of users

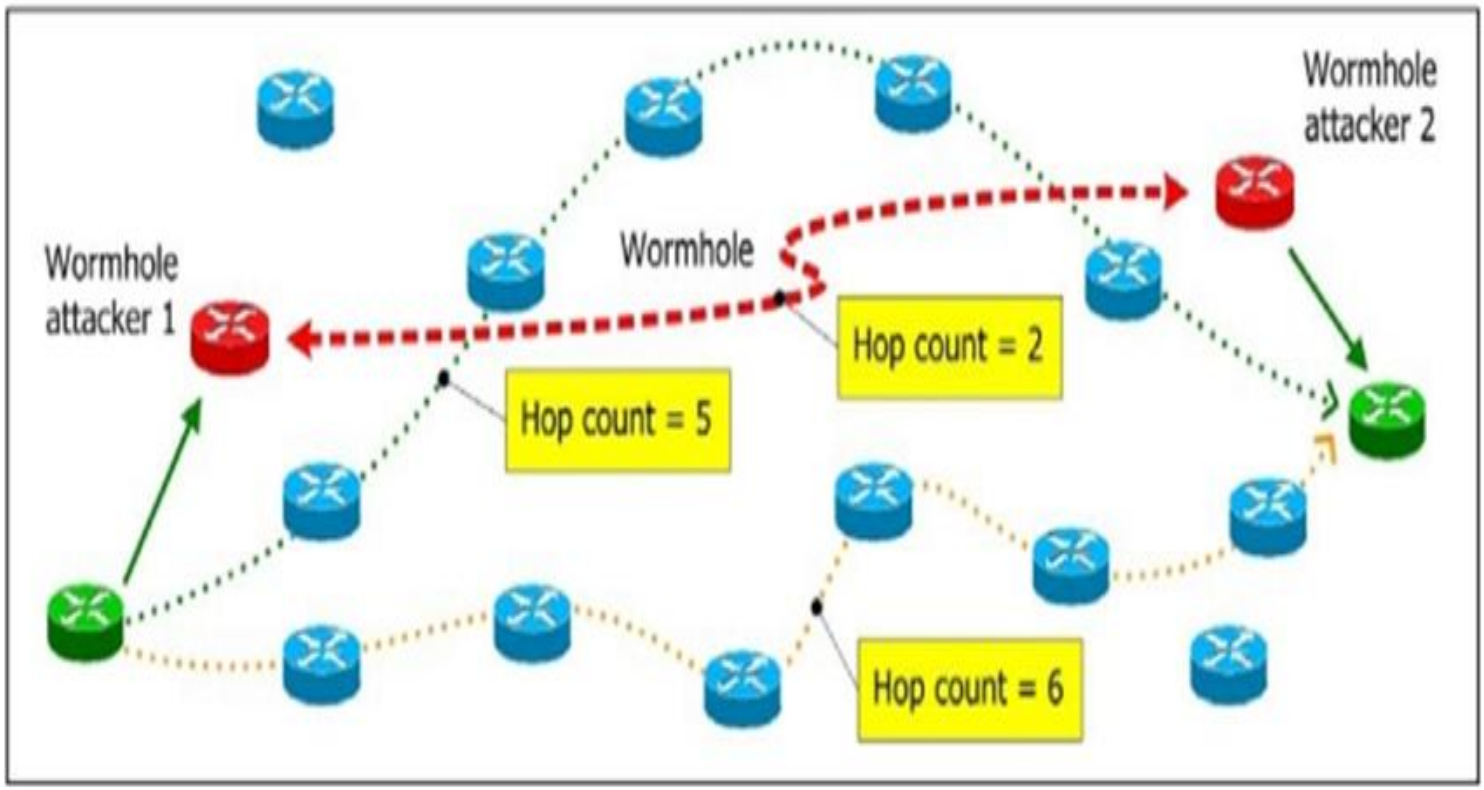
The concept is that no need to spend a lot of effort to catch thieves like the police (administrators), but rather lock the doors and windows as citizens (users), which is much easier and can avoid most of the threats with minimum effort. This method selects routes and "avoids" rather than "identify" the wormhole resulting in low cost and overhead. Typical example of this type is **Multipath Hop-count Analysis** (MHA, for short) to avoid wormhole attacks based on a *hop-count analysis* scheme. It is a highly efficient protocol which does not require any special supporting hardware. Furthermore, MHA is designed to use split multipath routes, so the transmitted data is naturally split into separate route. An attacker on a particular route cannot completely intercept (and subvert) our content [2].

Let us select one mitigation technique from the above list celled ***Multipath Hop-count Analysis (MHA)*** and explain how it works.

The researchers illustrate the concept of MHA in the following figure [2]. The normal routes for a particular communication pair contain 5~6 hops; the route under a wormhole attack has a hop-count of 2. The researcher sees the route under the wormhole attack has a smaller hop-count than

normal. As a result, users who avoid routes with relatively small hop-counts can avoid most wormhole attacks.

In MHA, the researchers first examine the hop-count values of all routes. Then researchers choose a safe set of routes for data transmission. Finally, researchers randomly transmit packets through safe routes. Even if the wormhole is not avoided in some severe cases, researchers can still minimize the rate of using the route path through the wormhole [2].



2. Session hijacking

Name of attack: session hijacking

Type of attack:

Type Basic classification	Type based on OSI layer	Type based on MANET(network layer)
Passive and Active attack	Transport and application layer	……

Date of attack: February 2006 G.C

Affected organization/computer: every interconnected computing device in the network specially those compute ***website*** between client and server

2.1. Working principle

The most popular criminals for carrying out a session hijacking are session sniffing, predictable session token ID, man in the browser, client-side and session fixation.

Session sniffing: it is one of the most basic techniques used with application layer session hijacking. The attacker uses a sniffer, such as Wire shark, or a proxy, such as OWASP Zed, to capture network traffic containing the session ID between a website and a client. Once the attacker captures this value, he can use this valid token to gain unauthorized access [5].

Predictable session id: many web servers use a custom algorithm or predefined pattern to generate session IDs. The greater the predictability of a session token, the weaker it is and the easier it is to predict. If the attacker can capture several IDs and analyze the pattern, he may be able to predict a valid session ID [5].

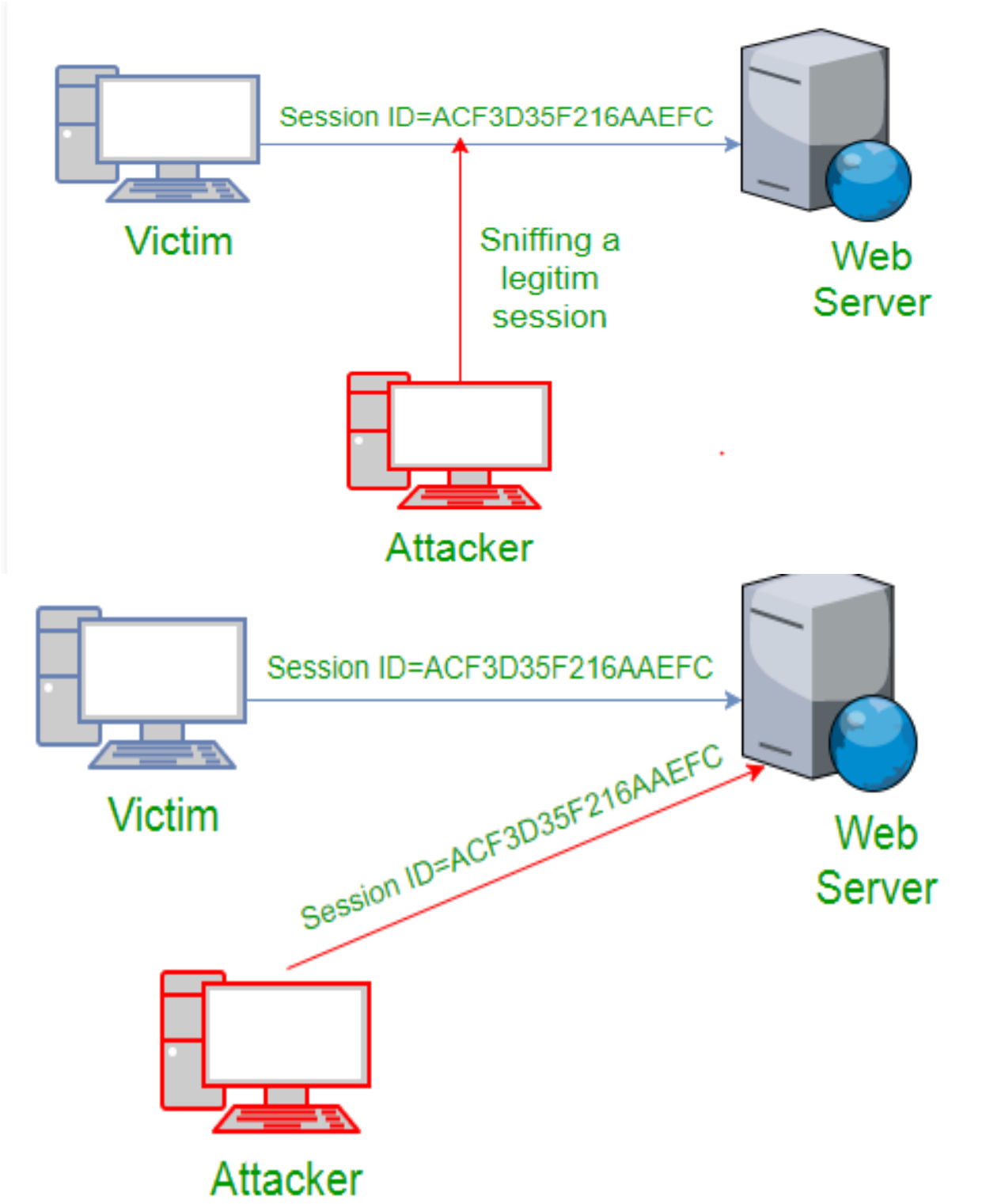
A man-in-the-browser attack: attacker first infects the victim's computer with a Trojan through some form of trickery or deceit. Once the victim is tricked into installing malware onto the system, the malware waits for the victim to visit a targeted site. The man-in-the-browser malware can invisibly modify transaction information and it can also create additional transactions without the user knowing. Because the requests are initiated from the victim's computer, it is very difficult for the web service to detect that the requests are fake [5].

Client-side attacks: it target end users. The idea is to extract session IDs stored on the victim's system or to use these tokens for the attackers benefit. Client-side attacks can include cross-site scripting (XSS), cross-site request forgery (CSRF) and malicious JavaScript codes [5].

Session fixation attacks: work by stealing a valid session ID that has yet to be authenticated. Then, the attacker tries to trick the user into authenticating with this ID. Once authenticated, the attacker now has access to the victim's computer. Session fixation explores a limitation in the way the web application manages a session ID. Three common variations exist [5]:

- Session tokens hidden in an URL argument
- Session tokens hidden in a form field and
- Session tokens hidden in a session cookie.

Generally, the session hijack attack is very **stealthy**. Session hijack attacks are usually waged against busy networks with a high number of active communication sessions. The high network utilization not only provides the attacker with a large number of sessions to exploit, but it can also provide the attacker with a **shroud** of protection due to the large number of active sessions on the server [5]. The following figure illustrates how session hijacking takes place [7]:


Session ID=ACF3D35F216AAEFC
Victim
Sniffing a legitim session
Web Server
Attacker
Session ID=ACF3D35F216AAEFC
Victim
Session ID=ACF3D35F216AAEFC
Web Server
Attacker

2.2. Mitigation options

Even though there is a lot of published solution to defend against wormhole attack, for our discussion we list out the following sample:

- SSL/TSL
- Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS)
- Transport layer defense method
- sequence number analysis
- received signal strength (RSS)
- received radio frequency finger printing (RFF)
- Round Trip Time values (RTT)
- Reciprocal Channel Variation based Identification (RCVI)
- One-Time Cookies

Let us select one mitigation technique from the above list celled ***one time cookie (OTC)*** and explain how it works.

OTC is designed to eliminate the vulnerabilities associated with the use of cookies as session authentication tokens. One-Time Cookies (OTC) is an HTTP session authentication protocol that is efficient, easy to deploy and resistant to session hijacking. OTC's security relies on the use of disposable credentials based on a modified hash chain construction. researcher implemented OTC as a plug-in for the popular Word Press platform and conducted extensive performance analysis using extensions developed for both Firefox and Firefox for mobile browsers. The experiments demonstrate the ability to maintain session integrity with a throughput improvement of 51% over HTTPS and a performance approximately similar to a cookie-based approach. In so doing, researchers demonstrate that one-time cookies can significantly improve the security of web sessions with minimal changes to current infrastructure [6].

OTC generates single-use authentication tokens based on a modified ***hash chain construction***. These tokens, once verified by the web application, cannot be reused. Moreover, each OTC credential is tied to a specific request for a resource, meaning that an adversary cannot intercept and repurpose them for illicitly redirecting a session [6].

Session integrity*:* OTC is more robust than authentication cookies and should prevent session hijacking attacks.

Performance*:* OTC has a minimal impact on the performance of the web application and the web browser.

Deployability: OTC is easy to implement in both, the web browser and the web application. It should have a low implementation cost.

Usability: OTC is easy to use and provide a user experience similar to the use of cookies.

The OTC protocol consists of two phases: setup and authentication.

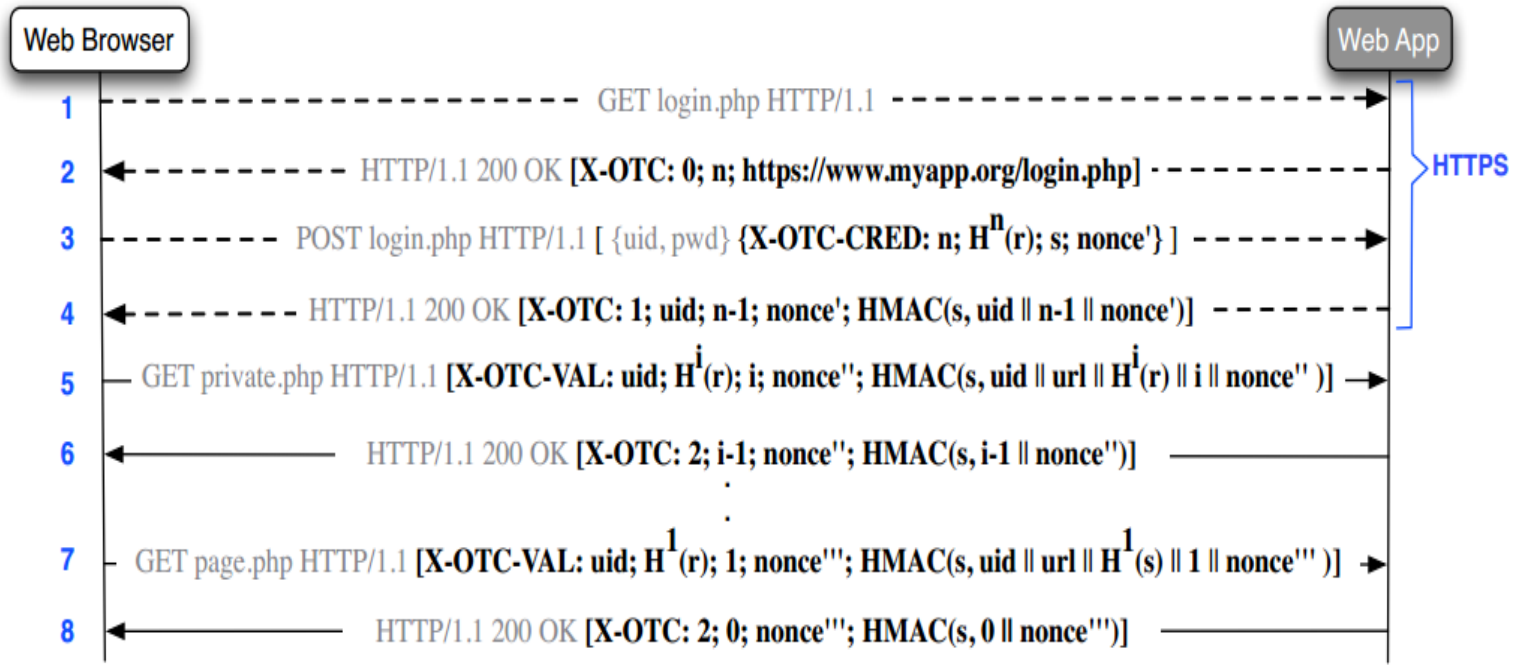
Setup phase

The web browser generates the OTC credentials and sends them to the web application. In the design, the browser generates the credentials to reduce the load in the web application, providing better scalability. Both the browser and the web application store OTC state. Once the server validates the OTC credentials, the authentication phase starts [6].

Authentication phase

During this phase, the browser uses the OTC credentials to generate one-time cookies that are appended to each request sent to the web application. These temporary credentials are then validated by the web application to verify the authenticity of the request. This phase continues until the hash chain in the OTC credentials is completely exhausted or the user ends her/his session [6].

The following diagram shows the Flow diagram of a web session authenticated with OTC. Messages 1 to 4 show the OTC setup phase and messages 5 to 8 show the OTC authentication phase. Each HTTP request and response includes an OTC header with protocol information. HMACs are used to protect the integrity of the values in the OTC headers and to tie each OTC value to a particular requested resource [6].


Web Browser
Web App
HTTPS
1 GET login.php HTTP/1.1
2 HTTP/1.1 200 OK [X-OTC: 0; n; https://www.myapp.org/login.php]
3 POST login.php HTTP/1.1 [{uid, pwd} {X-OTC-CRED: n; H^n(r); s; nonce'}]
4 HTTP/1.1 200 OK [X-OTC: 1; uid; n-1; nonce'; HMAC(s, uid || n-1 || nonce')]
5 GET private.php HTTP/1.1 [X-OTC-VAL: uid; H^i(r); i; nonce''; HMAC(s, uid || url || H^i(r) || i || nonce'')]
6 HTTP/1.1 200 OK [X-OTC: 2; i-1; nonce''; HMAC(s, i-1 || nonce'')]
7 GET page.php HTTP/1.1 [X-OTC-VAL: uid; H^1(r); 1; nonce'''; HMAC(s, uid || url || H^1(s) || 1 || nonce''')]
8 HTTP/1.1 200 OK [X-OTC: 2; 0; nonce'''; HMAC(s, 0 || nonce''')]

3. SQL injection

Name of attack: SQL injection

Type of attack:

Type Basic classification	Type based on OSI layer	Type based on MANET(network layer)
Active attack	Application layer	…….

Date of attack: February 2002 G.C

Affected organization/computer: computing device in the network those computing Web application related to database system

3.1. Working principle

SQL injection is a major concern when developing a Web application. It occurs when the application accepts a malicious user input and then uses it as a part of SQL statement to query a backend database.

An attacker can inject SQL control characters and command keywords (e.g., single quote (‘), double quote (**“**), equal (=), comment (- -), etc.) to change the query structure. Using these control characters with common SQL commands (e.g., SELECT, FROM, DELETE, UPDATE, etc.) enables access or retrieval of data elements from a backend database server [8].

A successful attack requires a Web application to include malicious code from an attacker in a SQL statement. The malicious code usually comes from an untrusted source. In some cases, internal system databases can also be the source of malicious data. When malicious SQL statements execute against a backend database, an attacker can modify or access the database. This depends how the attacker crafts the malicious data.

With a successful attack, an attacker can gain:

1. Unauthorized access to an application. An attacker can successfully bypass an application's authentication mechanism to have illegitimate access to it.
2. Information disclosure. A SQL injection attack could lead to a complete data leakage from the database server.
3. Loss of data availability. An attacker can delete records from the database server.
4. Compromised data integrity. As SQL statements are also used to modify or add the record, an attacker can use SQL injection to modify or add data stored in a database. This would lead to compromised data integrity.

Examples

The following line of code illustrates this vulnerability:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

This SQL code is designed to pull up the records of the specified username from its table of users. However, if the "userName" variable is crafted in a specific way by a malicious user, the SQL statement may do more than the code author intended. For example, setting the "userName" variable as:

```
' OR '1'='1
```

Or using comments to even block the rest of the query (there are three types of SQL comments. All three lines have a space at the end:

```
' OR '1'='1' --
' OR '1'='1' {
' OR '1'='1' /*
```

Renders one of the following SQL statements by the parent language:

```
SELECT * FROM users WHERE name = '' OR '1'='1';
SELECT * FROM users WHERE name = '' OR '1'='1' -- ';
```

3.2. Mitigation options

The manual for an SQL DBMS explains which characters have a special meaning, which allows creating a comprehensive blacklist of characters that need translation. For instance, every occurrence of a single quote (') in a parameter must be replaced by two single quotes ('') to form a valid SQL string literal. For example, in PHP it is usual to escape parameters using the function `mysqli_real_escape_string();` before sending the SQL query [8]:

```
$mysqli = new mysqli('hostname', 'db_username', 'db_password', 'db_name');
$query = sprintf("SELECT * FROM `Users` WHERE UserName='%s' AND
Password='%s'",
                  $mysqli->real_escape_string($username),
                  $mysqli->real_escape_string($password));
$mysqli->query($query);
```

This is one technique we can use while we write PHP script, the following mechanism also possible:

- Prepared statement with parameterized query
- Stored procedure
- Input validation
- Principle of least privilege
- Blacklist malicious host
- Pool resources
- Minimize access
- Encrypt data
- Distrust users
- Profile application
- Normalize input watch for automation

Next let us describe how protect against SQL injection ***one by one*** because each mitigation technique have ***minimized*** and ***short*** description by itself, so we hope that it does not become boring for read and understand .

Prepared statements with parameterized queries: Use of prepared statements with parameterized queries is a strong control to mitigate an attack. Instead of writing dynamic queries, which fails to differentiate between application code and data prepared statements force developers to use static SQL query and then pass in the external input as a parameter to query. This approach ensures the SQL interpreter always differentiates between code and data [9].

Stored procedures: Stored procedures are the SQL statements defined and stored in the database itself and then called from the application. Developers are usually only required to build SQL statements with parameters that are automatically parameterized. However, it's possible for a developer to generate dynamic SQL queries inside stored procedures. Implement stored procedures safely by avoiding dynamic SQL generation inside [9].

Input validation: A common source of SQL injection is maliciously crafted external input. As such, it's always a good practice to only accept approved input—an approach known as input validation [v].

Principle of least privilege: This is a standard security control that helps minimize the potential damage of a successful attack. Application accounts shouldn't assign DBA or admin type access onto the database server. Additionally, depending on access requirements, they should be restricted to least privileged access. For example, accounts that only require read access are only granted read access to the table they need to access. This ensures that if an application is compromised, an attacker won't have the rights to the database through the compromised application [9].

Blacklist malicious hosts: the most dangerous hosts can be identified, and then blacklisted against database access [10].

Pool resources: Businesses that share intelligence on SQL injection attacks could have a better picture of which hosts were launching such attacks. That said, according to Imperva, "the update rate of the blacklist must be high in order to keep up with new threats," because on average hosts only remain active for half a day [10].

3. Minimize access. Restrict the data that any given Web application can retrieve from a database. Never allow admin-level access to a database from a Web application [10].

Encrypt data: Never store data in plain text format. Rather, encrypt data, and at least salt and hash passwords, so that if attackers do manage to dump your database, they'll extract fewer pieces of high-value information [10].

Distrust users: "All input is evil." That's one essential Web application security mantra, according to Microsoft. What it means is that in an ideal scenario, Web application developers would only allow the inputs that they expect to receive, and would block all others [10].

Profile applications: Understand normal Web application behavior, so you can quickly identify when the application is behaving abnormally, such as attempting to execute many more database lookups than normal, or using unusual inputs [10].

Normalize inputs: Normalize database inputs "to avoid evasion attempts," said **Imperva**, then compare them against a database of known-bad inputs, to spot in-progress attacks [10].

Watch for automation: Since most SQL injection attacks are launched using automated tools, watch for indications of this technique. According to **Imperva**, "various mechanisms exist to detect usage of automatic clients, like rate-based policies and enforcement of valid client response to challenges [10] ".

References

Wormhole attack

[1]. Aniruddha B. Arnab B.and Dipayan B., different type of attack in mobile ADHOC network, prevention and mitigation technique, RETV 20/07/2011 E.C

[2]. Shang-Ming J., Chi-Sung L. and Wen-Chung K, a hop count analysis scheme for avoiding wormhole attacks in MANET, RETV 20/07/2011 E.C

[3]. Wang W., Bhargava B. Proceedings of the 2004 ACM workshop on Wireless Security (WiSe), ACM WiSE'04. Philadelphia, PA, USA: Oct, 2004. Visualization of Wormholes in Sensor Networks, RETV 20/07/2011 E.C

[4]. Lazos L., Poovendran R., Meadows C., Syverson P., Chang L.W. IEEE WCNC 2005. Seattle, WA, USA: 2005. Preventing Wormhole Attacks on Wireless Ad Hoc Networks: A Graph Theoretic Approach, RETV 20/07/2011 E.C

Session hijacking

[5]. what is session hijacking and how does it works, https://www.venafi.com/blog/what-session-hijacking, RETV 22/07/2011 E.C

[6]. ItaloD., SaurabhC., Mustaque A. and Patrick T, One-Time Cookies: Preventing Session Hijacking Attacks with Disposable Credentials, Georgia Institute of Technology, RETV 22/07/2011 E.C

[7]. Different way of session hijacking, https://www.geeksforgeeks.org/session-hijacking/, RETV 22/07/2011 E.C

SQL injection

[8]. what is SQL injection, https://en.wikipedia.org/wiki/SQL_injection, RETV 22/07/2011 E.C

[9]. Basic SQL injection and mitigation examples, https://www.geeksforgeeks.org/basic-sql-injection-mitigation-example, RETV 22/07/2011 E.C

[10]. Techniques to block SQL attack, https://www.darkreading.com/attacks-and-breaches/8-techniques-to-block-sql-attacks/d/d-id/1100239, RETV 22/07/2011 E.C