LFI (Local File Inclusion) কী?

LFI (Local File Inclusion) একটি সাধারণ ওয়েব অ্যাপ্লিকেশন দুর্বলতা যেখানে আক্রমণকারী ওয়েব সার্ভারে থাকা লোকাল ফাইলগুলোকে অ্যাপ্লিকেশনের মধ্য দিয়ে অ্যাক্সেস করতে পারে। এটি ঘটে যখন ওয়েবসাইট ইউজারের ইনপুট দিয়ে ফাইল ইনক্লুড করে, তবে সেই ইনপুট সঠিকভাবে ভ্যালিডেট করা হয় না। এর ফলে আক্রমণকারী সার্ভারের সিস্টেম ফাইল, কনফিগারেশন ফাইল, এমনকি কোড ফাইলও দেখতে পারে।

LFI কিভাবে কাজ করে?

LFI তখন ঘটে যখন একটি ওয়েব অ্যাপ্লিকেশন ইউজারের ইনপুট থেকে কোনো ফাইল ইঞ্জেক্ট করে এবং ফাইল পাথ যাচাই না করে সরাসরি ইনক্লুড করে। নিচে একটি সাধারণ উদাহরণ দেওয়া হলো:

```php
<?php
    $file = $_GET['page'];
    include($file);
?>
```

এই কোডে ইউজারের ইনপুট সরাসরি `include` ফাংশনে চলে যায়। যদি আক্রমণকারী নিচের মতো কোনো ইনপুট দেয়:

```
http://example.com/index.php?page=../../../../etc/passwd
```

তাহলে সার্ভার `/etc/passwd` ফাইলটিকে রেন্ডার করবে, যা আক্রমণকারীকে সিস্টেমের ইউজার ইনফরমেশন ফাইল দেখার সুযোগ দেবে।

Automated Tools LFI Vulnerabilities খুঁজতে:

LFI দুর্বলতা খুঁজতে ও এক্সপ্লয়েট করতে কিছু স্বয়ংক্রিয় টুল ব্যবহার করা যেতে পারে, যা প্রফেশনাল ও ইথিক্যাল হ্যাকারদের সময় বাঁচায়। নিচে কিছু জনপ্রিয় টুলের তালিকা এবং এগুলোর কাজের বর্ণনা দেওয়া হলো:

1. **Burp Suite**
   Burp Suite একটি ওয়েব পেনেট্রেশন টেস্টিং টুল যা LFI সহ অনেক ধরনের দুর্বলতা খুঁজতে ব্যবহার করা হয়। Burp Suite এর "Intruder" এবং "Repeater" মডিউল ব্যবহার করে LFI আক্রমণকে স্বয়ংক্রিয়ভাবে পরীক্ষায় ফেলা যায়।
   - **ব্যবহার:** আক্রমণের সময় ইউজার ইনপুটের মাধ্যমে বিভিন্ন ফাইল পাথ টেস্ট করা যায়, এবং Burp Suite রেসপন্সগুলি বিশ্লেষণ করে সম্ভাব্য LFI দুর্বলতা শনাক্ত করে।
2. **OWASP ZAP**
   OWASP ZAP (Zed Attack Proxy) একটি জনপ্রিয় ওয়েব অ্যাপ্লিকেশন সিকিউরিটি স্ক্যানার, যা LFI এবং অন্যান্য দুর্বলতাগুলি খুঁজে বের করার জন্য ব্যবহৃত হয়। এটি স্বয়ংক্রিয়ভাবে ওয়েবসাইট স্ক্যান করতে পারে এবং LFI ফাইলে প্রবেশের চেষ্টা করতে পারে।
   - **ব্যবহার:** ZAP দিয়ে অ্যাপ্লিকেশনের ফাইল ইনক্লুশন বিষয়গুলো স্ক্যান করে দুর্বলতা খুঁজে বের করা যায়।

3. **Commix**
   Commix একটি শক্তিশালী টুল যা কমান্ড ইনজেকশন দুর্বলতা খুঁজতে ব্যবহৃত হয়, কিন্তু এটি LFI এক্সপ্লয়েটেশনেও ব্যবহার করা যেতে পারে। এটি স্বয়ংক্রিয়ভাবে ফাইল ইনক্লুশন টেস্ট করতে পারে এবং অ্যাপ্লিকেশনে ম্যানুয়ালি ইনপুট দেওয়ার প্রয়োজন নেই।
   - ব্যবহার: ফাইল ইনক্লুশনের মাধ্যমে কমান্ড ইনজেকশন চেষ্টা করে আক্রমণকারী LFI এর সুযোগ নিতে পারে।
4. **Nikto**
   Nikto একটি ওপেন সোর্স ওয়েব সার্ভার স্ক্যানিং টুল, যা LFI দুর্বলতা সহ অনেক ধরনের সিকিউরিটি ইস্যু খুঁজে বের করতে পারে।
   - ব্যবহার: Nikto ওয়েবসাইটের বিভিন্ন প্যারামিটার পরীক্ষা করে দেখতে পারে যে, কোনো ইনপুট লোকাল ফাইল ইনক্লুশনের জন্য ঝুঁকিপূর্ণ কিনা।
5. **Fuzzing Tools (wfuzz, dirb)**
   wfuzz এবং dirb এর মতো ফাজিং টুল দিয়ে আক্রমণকারী URL প্যারামিটার থেকে ফাইল পাথগুলি স্বয়ংক্রিয়ভাবে পরীক্ষা করতে পারে। এগুলো সম্ভাব্য পাথ প্যাটার্ন ও ফাইল নাম দিয়ে টেস্ট করে LFI এক্সপ্লয়েট করতে সক্ষম হয়।
   - ব্যবহার: আক্রমণকারী বিভিন্ন ফাইল পাথ চেষ্টা করে দেখবে কোন কোন ফাইল ইনক্লুড করা যায়।

LFI থেকে বাঁচার উপায়:

- ইউজার ইনপুট স্যানিটাইজ করা।
- ইনপুট ফাইল নাম নির্দিষ্ট তালিকা (Whitelist) থেকে নির্বাচন করা।
- `realpath()` ফাংশন ব্যবহার করে ফাইলের সঠিক পাথ যাচাই করা।
- সঠিক ত্রুটি পরিচালনার মাধ্যমে আক্রমণকারীকে তথ্য দেওয়া থেকে বিরত থাকা।

এগুলো ব্যবহার করে LFI দুর্বলতাকে শনাক্ত এবং প্রতিরোধ করা সম্ভব।

LFI Vulnerability (Local File Inclusion), Server কীভাবে কাজ করে এবং কেন Linux কমান্ড ব্যবহার করা হয়?

LFI (Local File Inclusion) কীভাবে কাজ করে?

LFI মূলত ওয়েব অ্যাপ্লিকেশনে একটি ফাইল ইনক্লুশন দুর্বলতা, যেখানে অ্যাপ্লিকেশনটি ব্যবহারকারীর ইনপুট দিয়ে লোকাল ফাইল ইনক্লুড করে। এটি মূলত ঘটে যখন ইনপুট সঠিকভাবে যাচাই করা হয় না এবং ব্যবহারকারীর অনুরোধে ইনপুট করা ফাইলটি সরাসরি সার্ভার থেকে ইমপোর্ট হয়ে যায়।

উদাহরণস্বরূপ, ধরা যাক একটি ওয়েবসাইটে পেজ লোড করার জন্য নিচের URL ব্যবহার করা হয়:

```
http://example.com/index.php?page=about.php
```

এখানে `page` প্যারামিটার দিয়ে `about.php` ফাইল লোড করা হয়। যদি ইনপুট যাচাই না করা হয়, তাহলে আক্রমণকারী নিচের মতো অনুরোধ পাঠাতে পারে:

```
http://example.com/index.php?page=../../../../etc/passwd
```

এভাবে আক্রমণকারী সার্ভারের গুরুত্বপূর্ণ সিস্টেম ফাইল যেমন `/etc/passwd` দেখতে পারে, যা খুবই বিপজ্জনক।

Server কীভাবে কাজ করে?

ওয়েব সার্ভার মূলত একটি মিডলওয়্যার হিসেবে কাজ করে যা ক্লায়েন্ট (যেমন ব্রাউজার) এবং সার্ভারের মধ্যে যোগাযোগ স্থাপন করে। ওয়েব সার্ভার সাধারণত ক্লায়েন্টের অনুরোধ গ্রহণ করে, সেই অনুরোধ অনুযায়ী নির্দিষ্ট ফাইল বা ডেটা রেন্ডার করে এবং তারপর সেটি ক্লায়েন্টকে পাঠায়।

কিভাবে এটি কাজ করে**:**

1. ক্লায়েন্টের অনুরোধ: ক্লায়েন্ট একটি URL এর মাধ্যমে সার্ভারে রিকোয়েস্ট পাঠায়।
2. সার্ভার অনুরোধ গ্রহণ করে: ওয়েব সার্ভার (যেমন Apache, Nginx) রিকোয়েস্ট গ্রহণ করে এবং চায়না সেই রিকোয়েস্টটি কোন ফাইল বা রিসোর্সের জন্য।
3. ফাইল প্রসেস করা: সার্ভার রিকোয়েস্ট করা ফাইলটি খুঁজে পায় এবং অ্যাপ্লিকেশনের লজিক অনুযায়ী সেটা প্রোসেস করে।
4. রেসপন্স পাঠানো: প্রোসেস করা ফাইল বা আউটপুট ক্লায়েন্টের ব্রাউজারে পাঠানো হয়।

কেন **Linux** কমান্ড ব্যবহার করা হয়**?**

Linux সার্ভারগুলোতে কমান্ড লাইন ইন্টারফেস (CLI) অনেক বেশি শক্তিশালী এবং কার্যকরী, বিশেষ করে যখন সার্ভার ম্যানেজমেন্ট বা ওয়েব অ্যাপ্লিকেশন ডেপ্লয়মেন্টের কাজ করা হয়। কিছু গুরুত্বপূর্ণ কারণ নিচে দেওয়া হলো:

1. কার্যকারিতা **(Efficiency)**: Linux কমান্ড ব্যবহার করে সার্ভারের অনেক কাজ দ্রুত এবং সহজে করা যায়। GUI এর তুলনায় CLI অনেক বেশি হালকা ও দ্রুত।
2. স্বয়ংক্রিয়করণ **(Automation)**: অনেক ধরনের কাজ স্ক্রিপ্ট লিখে স্বয়ংক্রিয়ভাবে করা যায়। যেমন: ফাইল ম্যানেজমেন্ট, সার্ভার আপডেট, ব্যাকআপ ইত্যাদি কাজ Bash স্ক্রিপ্টিং দিয়ে সহজে করা যায়।
3. সিস্টেম কনফিগারেশন **(System Configuration)**: Linux সার্ভারগুলোতে সরাসরি কনফিগারেশন ফাইল এডিট করা প্রয়োজন হয়। `nano`, `vi` বা `vim` এর মতো টুলগুলো দিয়ে CLI থেকে খুব সহজে ফাইল এডিট করা যায়।
4. নিরাপত্তা **(Security)**: CLI কমান্ড ব্যবহার করে সার্ভার সিকিউরিটি ম্যানেজ করা সহজ এবং অধিকাংশ সিকিউরিটি টুল যেমন ফায়ারওয়াল, সার্ভিস মনিটরিং, লগ বিশ্লেষণ ইত্যাদি CLI থেকে নিয়ন্ত্রণ করা যায়।
5. প্রশস্ত কমিউনিটি সাপোর্ট: Linux কমান্ড লাইন এবং সার্ভার ম্যানেজমেন্টের ক্ষেত্রে অনেক টুল এবং সাপোর্ট আছে যা সিস্টেম অ্যাডমিনিস্ট্রেটরদের কাজকে সহজ করে দেয়।

Linux Command দিয়ে **LFI** এক্সপ্লয়েট করা

LFI দুর্বলতার সময় আক্রমণকারী Linux কমান্ড ব্যবহার করে সার্ভার থেকে গুরুত্বপূর্ণ ফাইল রিড বা এক্সিকিউট করতে পারে। নিচে উদাহরণ দেওয়া হলো:

LFI এর মাধ্যমে **`/etc/passwd`** ফাইল রিড করা:

```
http://example.com/index.php?page=../../../../etc/passwd
```

-

LFI এর মাধ্যমে **Apache** লগ ফাইল রিড করা এবং শেল কমান্ড ইনজেকশন:

```
http://example.com/index.php?page=/var/log/apache2/access.log
```

- এর মাধ্যমে আক্রমণকারী অ্যাপ্লিকেশন লগের মাধ্যমে শেল ইনজেক্ট করতে পারে, যা সার্ভারের সম্পূর্ণ নিয়ন্ত্রণ নিয়ে নিতে পারে।

প্যারামিটার ছাড়া **LFI (Local File Inclusion)** সম্ভব?

হ্যাঁ, প্যারামিটার ছাড়া **LFI (Local File Inclusion)** দুর্বলতা সম্ভব। সাধারণত LFI তখন ঘটে যখন ইউজারের ইনপুট সরাসরি URL প্যারামিটারের মাধ্যমে ফাইল ইনক্লুড করার জন্য ব্যবহৃত হয়। তবে এমন কিছু পরিস্থিতি আছে যেখানে প্যারামিটার ছাড়াও LFI দুর্বলতা থাকতে পারে। নিচে কিছু উদাহরণ দেওয়া হলো যেখানে প্যারামিটার ছাড়া LFI হতে পারে:

---

১. অ্যাপ্লিকেশনের হার্ডকোডেড পাথ

অনেক সময় অ্যাপ্লিকেশন এমনভাবে ডিজাইন করা থাকে যাতে ফাইল ইনক্লুশন করার জন্য ইউজারের ইনপুট না নিয়েও বিভিন্ন ফাইল লোড করা হয়। যদি এই ফাইল পাথ সঠিকভাবে নিরাপদ করা না হয়, তাহলে এটিও LFI দুর্বলতার কারণ হতে পারে।

```
<?php
    // ইউজারের লগইন অনুযায়ী একটি নির্দিষ্ট পেজ লোড করা হচ্ছে
    $user_role = $_SESSION['user_role'];

    if ($user_role == 'admin') {
        include('/pages/admin/dashboard.php');
    } else {
        include('/pages/user/home.php');
    }
?>
```

এই ক্ষেত্রে ফাইল ইনক্লুশন ইউজারের রোলের উপর ভিত্তি করে করা হচ্ছে, কোনো প্যারামিটার দিয়ে নয়। তবে যদি আক্রমণকারী কোনোভাবে সেশন ডাটা (**$_SESSION['user_role']**) পরিবর্তন করতে পারে (যেমন: সেশন হাইজ্যাকিং বা সেশন ফিক্সেশন), তাহলে সে সঠিক পাথ পরিবর্তন করে সার্ভার থেকে অন্য কোনো ফাইল ইনক্লুড করতে পারে।

---

২. Cookies বা Headers ম্যানিপুলেশন করে LFI

অনেক ওয়েব অ্যাপ্লিকেশন কুকি বা HTTP হেডার ব্যবহার করে ইউজারের ইনফরমেশন বা ফাইল সিলেকশন করে। যদি এই কুকি বা হেডার ইনপুট ভ্যালিডেট করা না হয়, তাহলে আক্রমণকারী সেটাকে ব্যবহার করে LFI আক্রমণ করতে পারে।

```
<?php
    // কুকি থেকে ইউজারের থিম সিলেকশন নেওয়া হচ্ছে
    $theme = $_COOKIE['theme'];
    include("themes/" . $theme . ".php");
?>
```

এই উদাহরণে, যদি `theme` কুকির ভ্যালু পরিবর্তন করে আক্রমণকারী `../../../../etc/passwd` দেয়, তাহলে সার্ভার গুরুত্বপূর্ণ সিস্টেম ফাইল ইনক্লুড করবে। এভাবে প্যারামিটার ছাড়াই আক্রমণকারী LFI দুর্বলতা ব্যবহার করতে পারে।

---

৩. Server-Side Request Forgery (SSRF) এর মাধ্যমে LFI

কিছু ক্ষেত্রে ওয়েব অ্যাপ্লিকেশন অভ্যন্তরীণ সার্ভিস বা API ব্যবহার করে ফাইল বা রিসোর্স রিকোয়েস্ট করে। যদি এই প্রক্রিয়া সঠিকভাবে সুরক্ষিত না হয়, তাহলে আক্রমণকারী এটাকে ব্যবহার করে LFI আক্রমণ করতে পারে।

```
<?php
    // একটি অভ্যন্তরীণ সার্ভিসে রিকোয়েস্ট পাঠানো হচ্ছে ফাইল কনটেন্ট পাওয়ার জন্য
    $content =
file_get_contents("http://internal-service/fetch?file=homepage");
    echo $content;
?>
```

যদি আক্রমণকারী এই রিকোয়েস্ট ম্যানিপুলেট করতে পারে এবং `file` প্যারামিটারে পাথ পরিবর্তন করতে পারে, তাহলে সার্ভারের অন্যান্য গুরুত্বপূর্ণ ফাইল ইনক্লুড করা সম্ভব হবে।

---

৪. File Upload Vulnerability এর মাধ্যমে LFI

ফাইল আপলোড দুর্বলতা থাকলে আক্রমণকারী এমন ফাইল আপলোড করতে পারে যা পরবর্তীতে ইনক্লুড করার জন্য LFI এর সুযোগ দেয়।

```
<?php
    // ফাইল আপলোড এবং ফাইল ইনক্লুশন
    $filename = $_FILES['file']['name'];
    move_uploaded_file($_FILES['file']['tmp_name'], "/uploads/" .
$filename);
    include("/uploads/" . $filename);
?>
```

এখানে আক্রমণকারী একটি ম্যালিসিয়াস PHP ফাইল আপলোড করে তারপর সেটি ইনক্লুড করতে পারে, ফলে সার্ভারে কোড এক্সিকিউশন সম্ভব হয়।