# NGINX

# PATH CONFIGURATION PITFALLS

The attack is based on specific conditions within the Nginx configuration.

# Nginx Path Configuration Pitfalls

At the BlackHat 2018 conference, Orange Tsai revealed a technique for exploiting URL parser misconfigurations. The attack is based on specific conditions within the Nginx configuration:

1. The `location` directive lacks a trailing slash in its path.
2. An `alias` directive is present within the `location` context, and it concludes with a slash.

**Understanding the Impact:**

In a vulnerable scenario, Nginx interprets URLs starting with `/img` and serves content from the specified alias path `/var/images/`. Notably, this means requests for both `/img/profile.jpg` and `/imgprofile.jpg` yield the same file. The absence of a trailing slash in the alias directive means an additional slash after the matched location is unnecessary.

**Exploiting the Vulnerability:**

Exploitation involves attempting to access the parent directory through a request URL starting with `/img`, for example, `/img..`. If Nginx responds with a redirection attempt to `/img../`, it signals the directory's detection.

This enables access to any file or child directory within the parent directory of the target folder. Exploiting this, an attacker can issue a GET request like `/img../log/nginx/access.log` to download sensitive log files, potentially leading to severe consequences.

**Impact Without a Slash on Alias:**

Even if the trailing slash on the alias directive is omitted, the vulnerability persists. Traversal sequences to escape the directory are no longer possible. However, the attacker can still exploit this behavior by accessing directories with names starting with the target directory name. For instance, while `/var/images/../log/` might be inaccessible, `/var/images_confidential` can be accessed through a GET request to `/img_confidential`.

In such cases, Nginx appends `_confidential` to the `/var/images` target path, serving URLs from the combined path of `/var/images` and `_confidential`, resulting in `/var/images_confidential`.

# Attacks

## Nginx Configuration Vulnerability

```
+-----------------------------------+
|            Nginx Server           |
|                                   |
|    location /img {                |
|        alias /var/images/;        |
|    }                              |
|                                   |
+-----------------------------------+
```

Illustrates the vulnerable Nginx configuration with the specified conditions.

## Exploiting Trailing Slash Misconfiguration

```
+-----------------------------------+
|            Nginx Server           |
|                                   |
|    Vulnerable URL Request:        |
|    /img..                         |
|                                   |
|    Response:                      |
|    301 Redirect to /img../        |
|                                   |
+-----------------------------------+
```

Demonstrates the exploitation of the trailing slash misconfiguration, triggering a redirection attempt.

## Exploiting Parent Directory Access

```
+-----------------------------------+
|            Nginx Server           |
|                                   |
```

```
|    Exploited URL Request:          |
|    /img../log/nginx/access.log     |
|                                    |
|    Response:                       |
|    Serves log file content         |
|                                    |
+------------------------------------+
```

Shows the exploitation of the vulnerability, accessing the parent directory's log file.

## Impact Without Trailing Slash on Alias

```
+------------------------------------+
|            Nginx Server            |
|                                    |
|    Exploited URL Request:          |
|    /img_confidential               |
|                                    |
|    Response:                       |
|    Serves content from             |
|    /var/images_confidential        |
|                                    |
+------------------------------------+
```

Depicts the impact even without a trailing slash on the alias, accessing a directory with a specific name.

## Combined Impact

```
+------------------------------------+
|            Nginx Server            |
|                                    |
|    Combined Exploitation:          |
|    /img../log/nginx/access.log     |
|    /img_confidential               |
|                                    |
|    Response:                       |
|    Serves log file and content     |
```

```
|    from /var/images_confidential   |
|                                     |
+-------------------------------------+
```

Combines the different aspects of the attack scenario, highlighting the potential for serving log files and content from specific directories.

# Defend Against Attacks

## 1. Update Nginx:

Keep Nginx up to date with the latest version.

```
sudo apt update sudo apt upgrade nginx
```

## 2. Configuration Check:

Always check the Nginx configuration for syntax errors before restarting.

`sudo nginx –t`

## 3. Use Configuration Management:

Utilize configuration management tools for managing Nginx configurations.

## 4. Security Headers:

Implement security headers in the Nginx configuration file.

```
location /restricted {
    deny all;
    return 403;
}
```

## 5. Access Control:

Restrict access to specific locations.

```
location /restricted {
    deny all;
    return 403;
}
```

### 6. Directory Listing:

Disable directory listing.

```
location / {
    autoindex off;
}
```

### 7. Alias Traversal Protection:

Use the `alias` directive carefully to prevent directory traversal.

```
location /files/ {
    alias /path/to/files/;
}
```

### 8. HTTP to HTTPS Redirect:

Redirect HTTP traffic to HTTPS.

```
server {
    listen 80;
    server_name example.com;
    return 301 https://$host$request_uri;
}
```

### 9. SSL Configuration:

Configure SSL for HTTPS.

```
server {
    listen 443 ssl;
    server_name example.com;
```

```
    ssl_certificate /path/to/certificate.crt;
    ssl_certificate_key /path/to/private.key;
    # Other SSL configurations
}
```

## 10. Rate Limiting:

Implement rate limiting to prevent abuse.

```
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;
server {
    location / {
        limit_req zone=one burst=5;
        # Other configurations
    }
}
```

## 11. Connection Limits:

Set connection limits to protect against DDoS attacks.

```
events {
    worker_connections 1024;
}
```

## 12. Custom Error Pages:

Customize error pages for a better user experience.

```
error_page 404 /404.html;
```

## 13. Gzip Compression:

Enable Gzip compression for better performance.

```
gzip on;
gzip_types text/plain text/css application/json application/javascript;
```

## 14. Client-Side Caching:

Implement client-side caching.

```
location ~* \.(jpg|jpeg|png|gif|ico|css|js)$ {
    expires 7d;
}
```

## 15. HTTP2 Protocol:

Enable HTTP/2 for improved performance.

```
listen 443 ssl http2;
```

## 16. Secure File Permissions:

Ensure secure file permissions.

```
chmod 644 /path/to/file
```

## 17. Web Application Firewall (WAF):

Consider using a WAF for additional security.

## 18. Monitoring and Logging:

Set up monitoring and logging.

```
access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log;
```

## 19. SSH Hardening:

Harden SSH access to the server.

`nano /etc/ssh/sshd_config`

## 20. Firewall Configuration:

Configure a firewall to allow only necessary traffic.

```
sudo ufw allow 80
sudo ufw allow 443
sudo ufw enable
```

### 21. Two-Factor Authentication:

Enable two-factor authentication for server access.

### 22. Regular Backups:

Implement regular backups of configurations and data.

### 23. Deny Hidden Files:

Deny access to hidden files.

```
location ~ /\. {
    deny all;
}
```

### 24. IP Whitelisting:

Whitelist specific IP addresses.

```
location / {
    allow 192.168.1.1;
    deny all;
}
```

### 25. Disable Unused Modules:

Disable unnecessary Nginx modules to reduce potential attack surfaces.

```
# Comment out or remove unnecessary modules
# load_module modules/ngx_http_ssi_filter_module.so;
```

### 26. Use Trailing Slash in Alias Directives:

Ensure that the alias directive in the location block ends with a trailing slash. This helps prevent directory traversal vulnerabilities. For example:

```
location /assets/ {
    alias /opt/production/assets/;
}
```

### 27. Regular Expression Matching:

If possible, consider using regular expression matching in the location directive. Regular expression matching provides more control over URL patterns and can enhance security.

```
location ~ ^/assets/ {
    alias /opt/production/assets/;
}
```

### 28. Implement Strict Location Paths:

Be specific in defining location paths to minimize the risk of unintended matches. Avoid using broad location blocks that can expose sensitive directories.

```
location /assets/ {
    alias /opt/production/assets/;
}
```

---

Discord: https://discord.gg/CqV6aJXMkA

Telegram: https://t.me/Hadess_security

# Reference


- https://labs.hakaioffsec.com/nginx-alias-traversal/
- https://devsecopsguides.com/docs/checklists/nginx/