CSE 4531

Server Side Vulnerabilities

Path traversal / Directory traversal

Enable an attacker to read arbitrary files on the server that is running an application.

- Application code and data.
- Credentials for back-end systems.
- Sensitive operating system files.

In some cases, an attacker might be able to write to arbitrary files on the server, allowing them to modify application data or behavior, and ultimately take full control of the server.

Continued

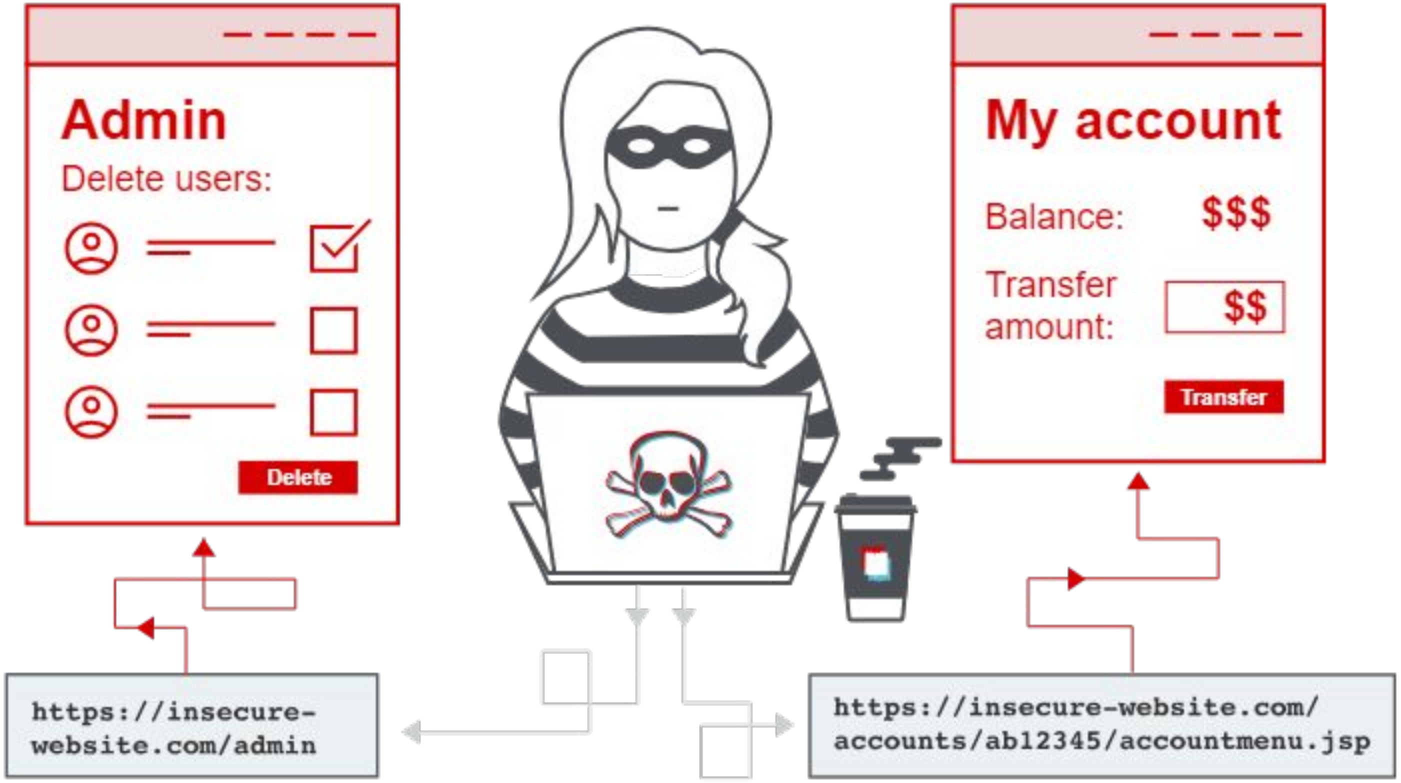
- **`<img src="/loadImage?filename=218.png">`**
- **`/var/www/images/218.png`**
- **`https://insecure-website.com/loadImage?filename=../../../etc/passwd`**
- **`https://insecure-website.com/loadImage?filename=..\..\..\windows\win.ini`**

[LINKforLAB]

Access Control

- determines whether the user is allowed to carry out the action that they are attempting to perform.
- Authentication and Session management.
- Vertical and Horizontal privilege escalation.

Illustration of Broken Access Control


Admin
Delete users:
Delete
My account
Balance: $$$
Transfer amount: $$
Transfer
https://insecure-website.com/admin
https://insecure-website.com/accounts/ab12345/accountmenu.jsp

Vertical Privilege Escalation

- We use different files to help crawlers.
- `https://insecure-website.com/admin`
- `https://insecure-website.com/robots.txt`
- wordlist to brute-force
- LINKforLAB
- Solution can be using uncommon url extension like :

  `https://insecure-website.com/administrator-panel-yb556`

Continued (New Problem)

- The application might still leak the URL to users.
- Observe the below piece of code,

```
<script>

var isAdmin = false;

if (isAdmin) {

...

var adminPanelTag = document.createElement('a');
adminPanelTag.setAttribute('https://insecure-website.com/administrator-panel
-yb556');

adminPanelTag.innerText = 'Admin panel';

...

} </script>
```

- LinkForLab

Parameter Based Access Control Vulnerabilities

- The application makes access control decisions based on the submitted value (cookie,hidden field),

```
https://insecure-website.com/login/home.jsp?admin=true
https://insecure-website.com/login/home.jsp?role=1
```

- What is the problem?
- LinkForLab
- Solution : Don’t use forgeable cookie or anything user can change.

Horizontal Privilege Escalation

- IODR vulnerability. (access resources not rights)
- Can access another user by incrementing one’s id,

```
https://insecure-website.com/myaccount?id=123
```

- Solution GUID. This may prevent an attacker from guessing or predicting another user's identifier.
- But GUID must be kept secret without disclosing it anywhere.
- LinkForLab
- HorizontalToVertical : Using Horizontal method to access admin credentials.

Authentication

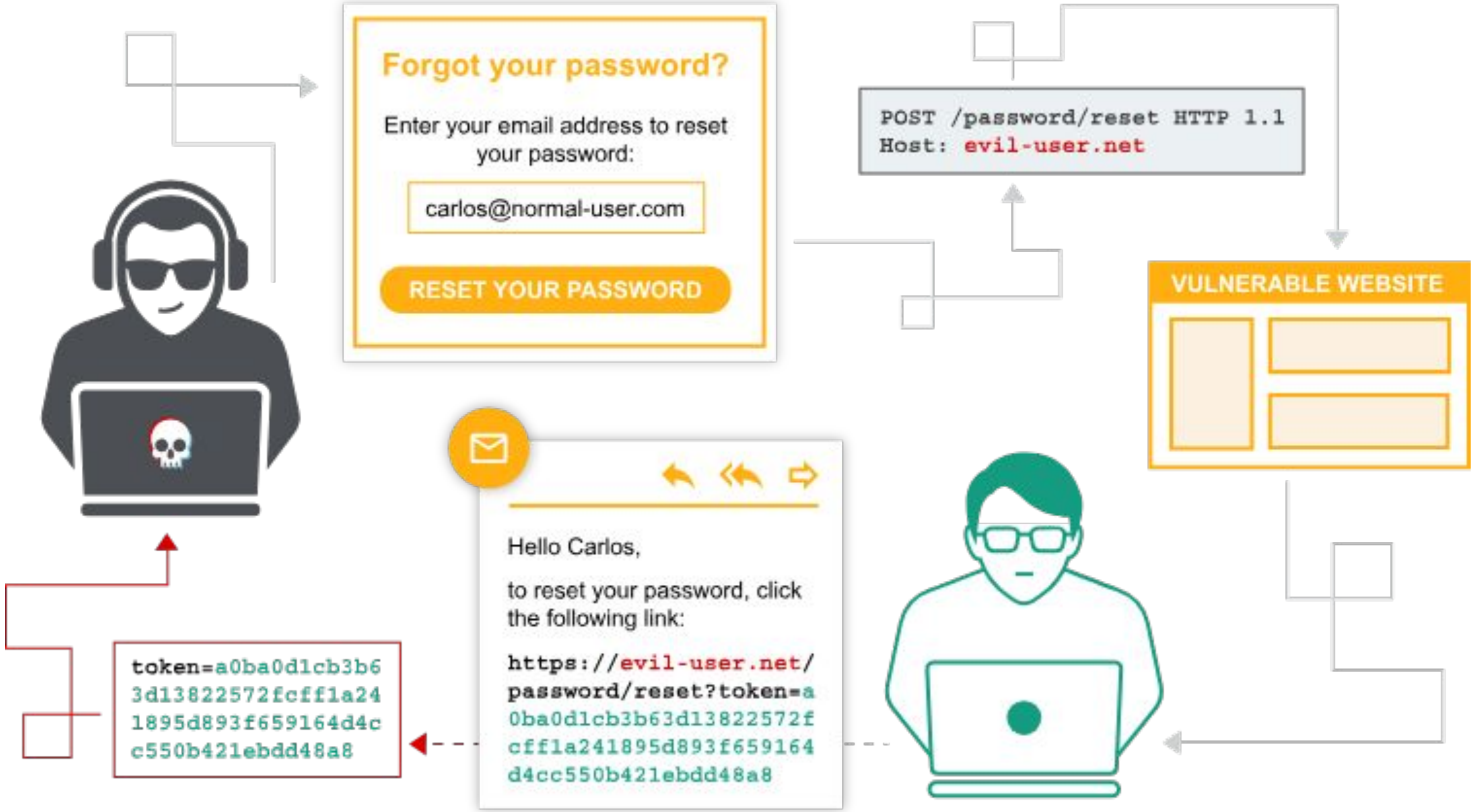
- Authentication VS Authorization
- Brute Force attacks
    - Using tools to automate the process.
    - Can use word list
    - Fine tune the process with basic logic and publicly available knowledge like birthday.
- Brute Forcing Usernames
    - `firstname.lastname@somecompany.com`
- Brute Forcing Passwords
    - Website Forcing us to write high entropy passwords.
    - But what we do is convert `mypassword -> Mypassword1!`

Conclusion, brute force is not simply iterating all combinations of possibles.

Continued

- Username Enumeration is observing website’s behavior to identify validity of a username.
    - Status Codes
    - Error Messages
    - Response Times
- LinkForLab
- Solution two-factor authentication
- Bypassing two-factor authentication
    - User is already in a “logged in state” before entering the verification code.
    - Check whether we can directly access user page without performing verification.
- LinkForLab

Illustration of Authentication Vulnerability



Forgot your password?

Enter your email address to reset your password:

carlos@normal-user.com

```
POST /password/reset HTTP 1.1
Host: evil-user.net
```

Hello Carlos,

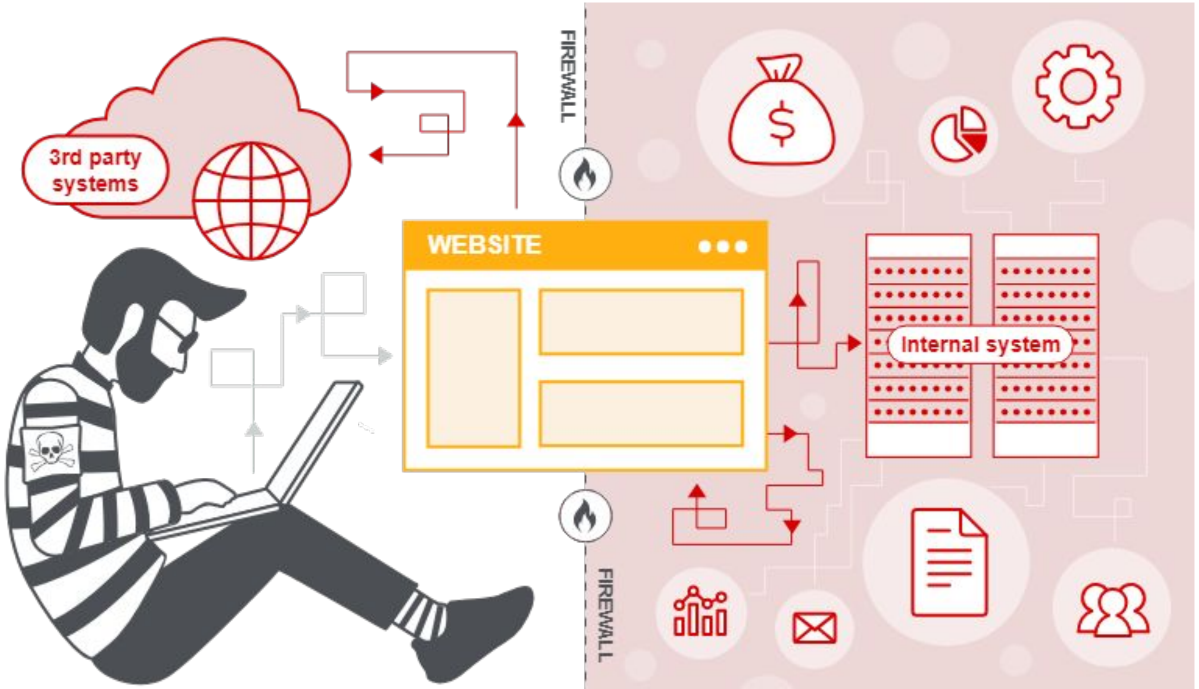
to reset your password, click the following link:

```
https://evil-user.net/
password/reset?token=a
0ba0d1cb3b63d13822572f
cff1a241895d893f659164
d4cc550b421ebdd48a8
```

```
token=a0ba0d1cb3b6
3d13822572fcff1a24
1895d893f659164d4c
c550b421ebdd48a8
```

SSRF

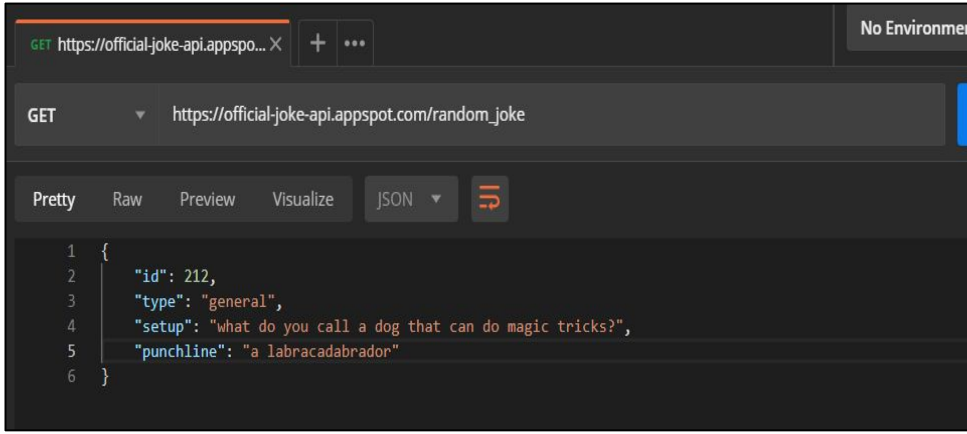
An attacker might force server-side application to make request to unwanted external system and compel server to disclose authorization credentials.



Continued (REST API)

Before learning more about SSRF, we should learn more about REST API,

- REST(Representational state transfer) defines set of rules for communication between a client and server (sometimes a server can be a client to another server).
- Example,
  - A client ask a server for a certain data and server response with the queried data.
  - Now REST defines how the client is going to ask and how the server will respond.
- There is an ENDPOINT (URL) where server is waiting for our request, and at that point we dump our query and some information about us using which server can respond to us.

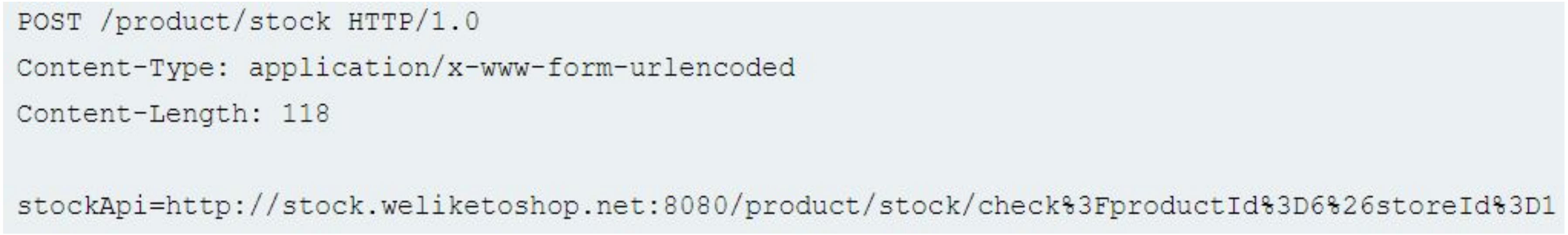


GET https://official-joke-api.appspot.com/random_joke

```
{
    "id": 212,
    "type": "general",
    "setup": "what do you call a dog that can do magic tricks?",
    "punchline": "a labracadabrador"
}
```

Continued (Back to SSRF)

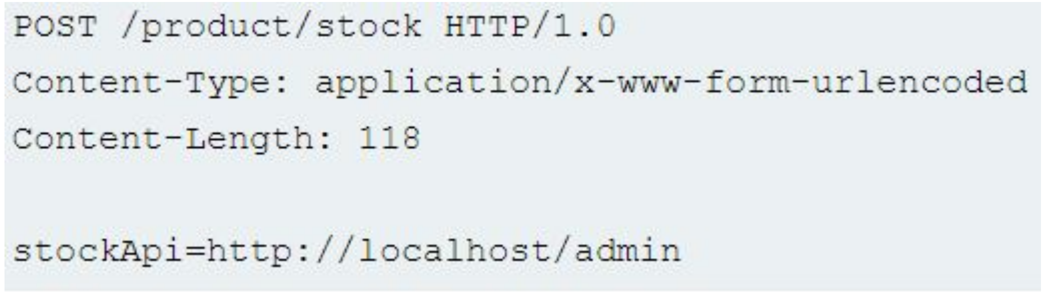
If a user ask a server to tell him/her the amount of product available at the stock, server might need to communicate with a REST API. In the below image, server is asking API for the stock value,



```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

stockApi=http://stock.weliketoshop.net:8080/product/stock/check%3FproductId%3D6%26storeId%3D1
```

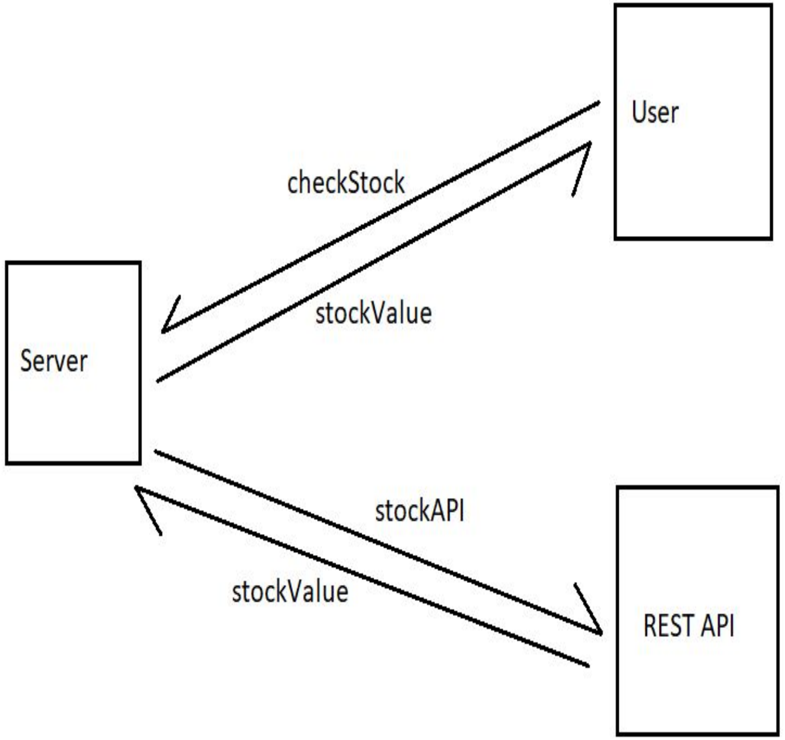
The right side image depicts where an attacker intercept the above request and change it so that he/she can get the raw data of admin page. Raw data means you can view it but can't expect it to be a running website.



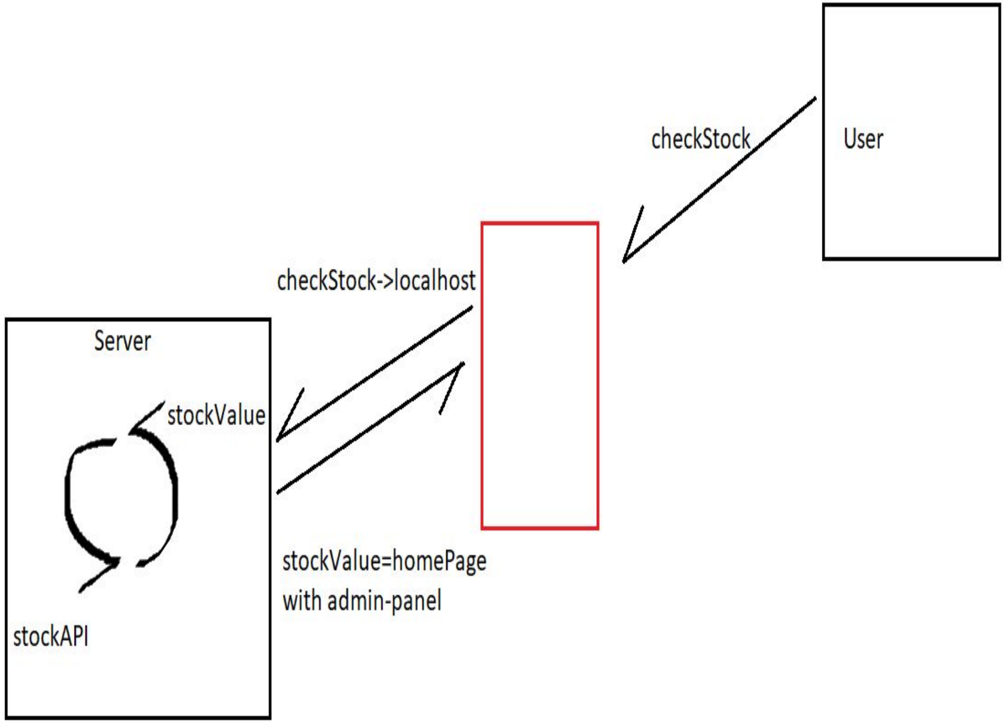
```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118

stockApi=http://localhost/admin
```

Illustration of SSRF



Normal REST communication



SSRF

Continued (Trust Issues)

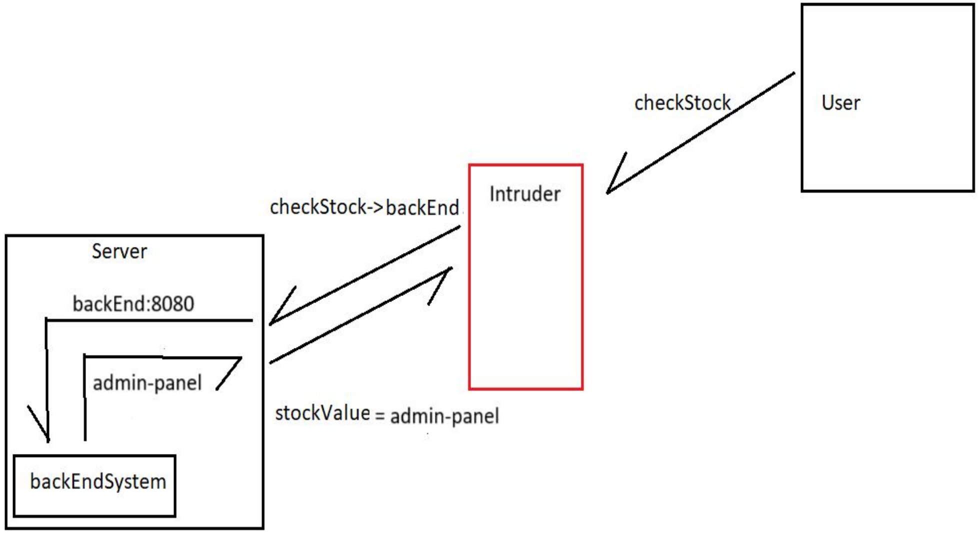
Why do applications implicitly trust requests that come from the local machine?

- Position of Access Control Component
- Disaster Recovery Purposes
- Different Port Number

LinkForLab

Not only external system attacker can target internal system located at different port number.

LinkForLab



File Upload Vulnerability

- An attacker give a file to web server and without proper validation (like checking its size,content or type) it uploads that file to its file system.
- This file could be server-side script file that enable remote code execution.
- Sometimes just uploading will start the attack.
- In other cases attacker do a HTTP request to retrieve/view that file and our web server will execute that file without knowing it is a sort of computer virus.

Continued

Developers try their best to shield their website from file upload vulnerabilities but,

- Check file type but not extension.
- Compare with blacklist but there is always some file type which is not included in the blacklist.
- Check file type using properties which can be manipulated by burp suite or any other equivalent tools.
- Inconsistencies in applying file validation through out the network and directories.

Some important definitions,

- Web Shell : A web shell is a malicious script that enables an attacker to execute arbitrary commands on a remote web server simply by sending HTTP requests to the right endpoint.
- Pivot Attack : After accessing one web server, using that attacker takes control of subsequent web servers in the same organisation or network.

Continued (Exploiting File Upload Vulnerabilities)

Detail process of exploiting file upload vulnerabilities,

- We upload (using POST) a file named exploit.php but the file type would be image/png (using burp). This file contains this piece of code

```
“<?php echo file_get_contents('/home/carlos/secret'); ?>”
```

- Wherever exploit.php might be saved the file_get_contents method always assume that it is in the root directory. So it will take us from the root to carlos folder and get contents of secret folder/file. If it is a folder then it will return us a list of file and if it is a file, it will give us what ever written inside our file.
- Now to run exploit.php we give a GET request stating the path directory of our exploit.php like /avatar/exploit.php again assuming we are already in our root directory. The request look like this,

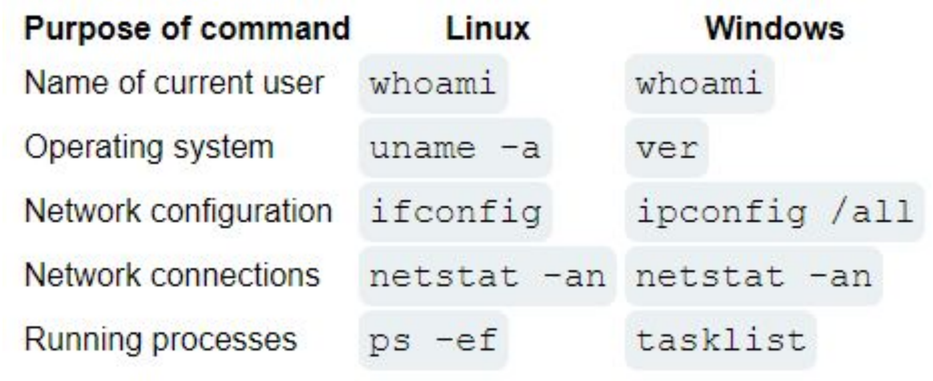
```
“GET /files/avatars/exploit2.php HTTP/2”
```

- After sending the above GET request as a response server will execute exploit.php and for php echo command we get to see contents of carlos’s secret file. LinkForLab
- Remember, in the above process the required vulnerability is that developer checked type of the file but didn’t check the file extension. LinkForLab

Os command/shell injection

Allows an attacker to execute operating system (OS) commands on the server that is running an application, and typically fully compromise the application and its data.

Both shell injection and exploiting upload vulnerabilities can be used to start a pivot attack.



Purpose of command	Linux	Windows
Name of current user	`whoami`	`whoami`
Operating system	`uname -a`	`ver`
Network configuration	`ifconfig`	`ipconfig /all`
Network connections	`netstat -an`	`netstat -an`
Running processes	`ps -ef`	`tasklist`

Some important os commands for attackers.

Continued (Process)

- Let’s say one website doesn’t check what sort of parameter is going through its interface and finally to it’s server.
- If an attacker get hold of a POST request using burp they can change vulnerable parameter to a shell command like ‘whoami’. Like,
- productId=1&storeId=1 -> productId=&whoami #&storeId=1. Here & means command separator and # means comment out in OS.
- But attacker need to tweak and modify the above request a little. Because the first ‘&’ and second ‘&’ is not same. The red colored phrase is OS command and other part of the above command is format for REST communication. So to convert OS command to REST we need to encode it. After encoding it will look like, productId=%26whoami+%23&storeId=1
- Now if we send this POST request, server will response with the user information.

SQLi

Allows an attacker to interfere with the queries that an application makes to its database. Using SQLi attackers can,

- View, update or delete data of other users.
- Escalate from this to compromise server or any back end infrastructure.
- Perform denial-of-service.

Some ways to detect SQLi vulnerabilities by submitting,

- Single quote character
- OR 1=1

Continued (Simple Exploit)

- Normally this below url `https://insecure-website.com/products?category=Gifts`

  Means this : `SELECT * FROM products WHERE category = 'Gifts' AND released = 1`

- The sql command is going to show every products which are categorized as gifts and already released.
- Now let’s say we do this `https://insecure-website.com/products?category=Gifts'--`

  Means this : `SELECT * FROM products WHERE category = 'Gifts'--' AND released = 1`

- After ‘--’ everything is commented out. So showing products which has yet to be released.
- Again : `https://insecure-website.com/products?category=Gifts'+OR+1=1 --`

  Means : `SELECT * FROM products WHERE category='Gifts' OR 1=1--' AND released=1`

- Gifts or not showing every products in the table.
- Even as an attacker we should be careful about ‘OR 1=1’ injection since it might result in data loss.
- We can use this same logic and can access to server (which has SQLi vulnerability) as administrator. This is called escalating from SQLi to other attacks.