# Security incident report

| **Section 1: Identify the network protocol involved in the incident** | |
| --- | --- |
| The network protocol involved in the incident is the **Hypertext Transfer Protocol (HTTP)**. This is identified in the tcpdump traffic log by connections initiated on port **80**, which is the default port for HTTP communication. The client system established a connection with the website yummyrecipesforme.com and later with greatrecipesforme.com using this protocol. Since HTTP operates at the **Application Layer** of the **TCP/IP model**, it was used to transmit the payload that included the malicious executable and triggered a redirection to a spoofed website. | |
|

| **Section 2: Document the incident** |
| --- |
| On the day of the incident, multiple users reported to the helpdesk that while accessing the website yummyrecipesforme.com, they were unexpectedly prompted to download a file in order to access free recipes. After executing the downloaded file, users observed that the browser redirected them to another site, greatrecipesforme.com, and that their computers began to perform unusually slow.  In response, the website owner attempted to log into the administrative panel but was unable to do so. The incident was escalated to the cybersecurity team for further investigation.  To safely analyze the event, a **sandbox environment** was created. Using a **network protocol analyzer (tcpdump)**, traffic was captured during the browsing session to reproduce the user experience. The captured logs confirmed the following sequence:   * A **DNS request** was made by the browser to resolve the IP address of yummyrecipesforme.com. * A valid **DNS response** was received. * A **TCP three-way handshake** (SYN, SYN-ACK, ACK) was completed between the client and the server. * The client issued an **HTTP GET** request to the site. * A file download prompt appeared and the file was executed within the sandbox. * Following this, a new **DNS query** was issued for greatrecipesforme.com. * Another **HTTP connection** was made to the new domain, confirming redirection.   Upon deeper inspection by a senior analyst, it was discovered that a **JavaScript function** was injected into the website’s source code. This function triggered the file download and redirected users to the spoofed domain.  Further investigation confirmed that the attacker, a disgruntled ex-employee, gained unauthorized access to the admin panel using a **brute force attack**. The attacker repeatedly attempted default passwords and was able to access the panel because the **administrator account was still using the default password**. The attacker then modified the site’s code and changed the admin credentials, locking out legitimate access.  This sequence of events, user reports, network logs, and source code analysis confirmed that the attacker compromised the web server through **weak password security and the absence of brute force protection mechanisms**. |

| **Section 3: Recommend one remediation for brute force attacks** |
| --- |
| To prevent future brute force attacks, it is recommended to implement **Two-Factor Authentication (2FA)** on all administrative accounts. 2FA adds an extra layer of security by requiring a secondary verification step—such as a code sent via SMS, email, or an authenticator app—after the password is entered.  Even if an attacker guesses or obtains the correct password, they will not be able to gain access without the second authentication factor. This dramatically reduces the chances of unauthorized access through brute force attempts and enhances overall account security. |