**Simulated Phishing Attack and Forensic Response: A Report on Incident Response**

**Executive Summary:**

In our project, we simulated a phishing attack that was carried out in the form of having a malicious resume attachment via an email to the HR department of a fictional company, Quick Logistics LLC. When the resume is opened, embedded malicious VBA macros trigger the downloading of an additional payload through a JavaScript file that aims to do further damage to the end user.

This report aims to analyze the attacker’s tactics and identify the precise areas of compromise within the system in this simulation. The simulation is inspired by TryHackMe’s Boogeyman 2 challenge with changes made based on the project requirements.

The key steps taken in our analysis include the collection of artefacts (i.e., email headers, memory dumps, etc.) and the subsequent analysis we carried out on these artefacts using tools such as Olevba, Volatility, and inspection of suspicious URLs.

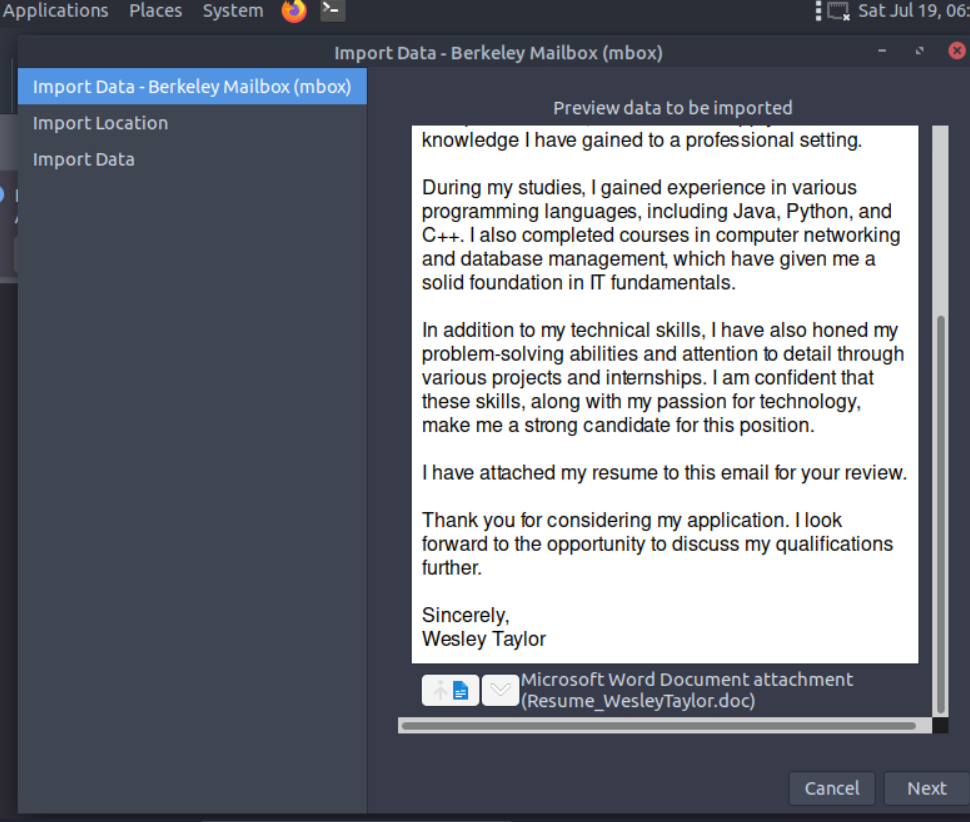
With the use of the aforementioned forensic tools, we found the phishing email contained a malicious Word document (Resume.doc) embedded with VBA macro code. When the victim opened the document, the macro automatically executed. The execution of VBA code in the resume doc triggered the download of a secondary JavaScript payload (update.js) from a remote server. This script was then executed using wscript.exe. By leveraging wscript.exe, the attacker was able to bypass basic security controls and initiate further compromise of the system without raising immediate suspicion.

Based on our analysis of the attacker’s tactics in the simulation, we came up with an incident-response playbook that incorporated the lessons learned from this simulation in order to provide guidance on how to address similar future incidents.

**Overview of Incident Simulation:**

The event simulated a phishing email that was directed at an employee of Quick Logistics LLC, which posed as a job applicant. The email had a malicious Word document attached. (Resume.doc) that contained dangerous VBA (Visual Basic for Applications) macros. When the recipient opened the document, the macros downloaded a JavaScript file (update.js) from an attacker-controlled remote server.

The JavaScript file (update.js) was then executed using the trusted Windows scripting host wscript.exe. This method allowed the attackers to bypass typical security defenses and gain initial access to the employee’s computer.



Once completed, the JavaScript connected with a distant command-and-control (C2) server to get extra instructions and payloads for maintaining persistent control of the compromised machine and enabling potential further data exfiltration from the compromised client machine.

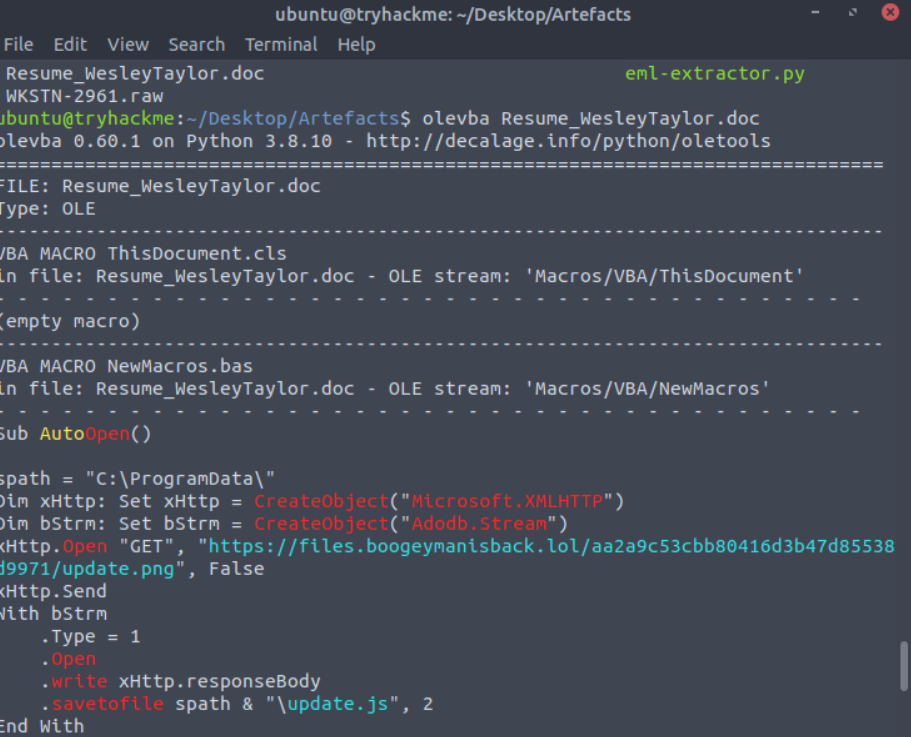
**Evidence Collection:**

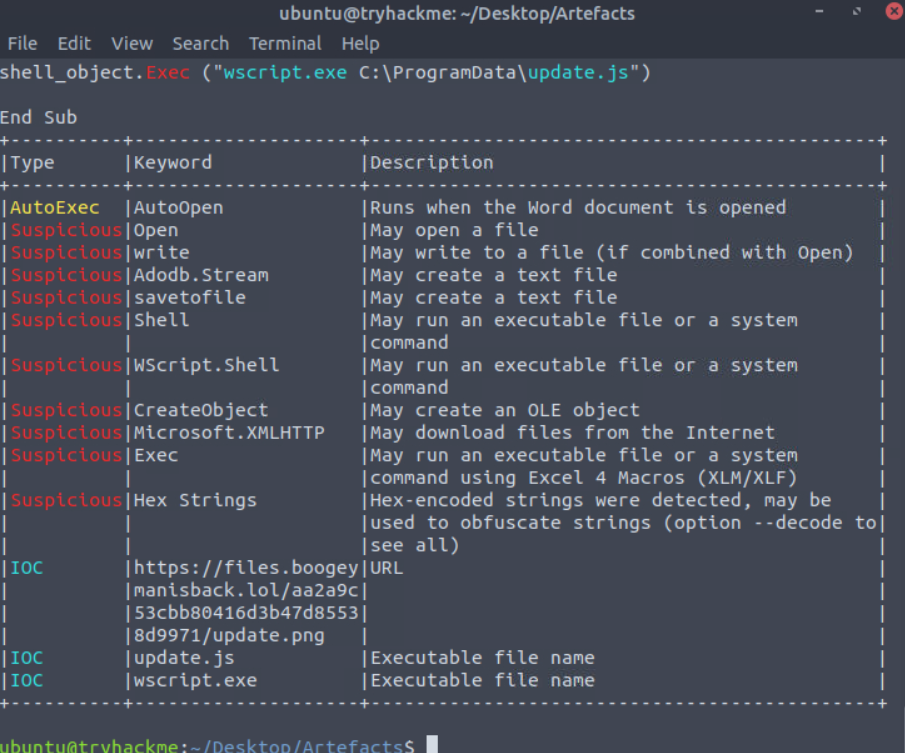
We gathered several digital artifacts from the breached environment to thoroughly analyze the simulated phishing attack in order to analyze the specific tactics used by the attacker to gain initial access to the HR employee’s computer.

We analyzed the email headers from the phishing email that had been dispatched to the HR division. By analyzing the header, we were able to get our hands on the metadata like sender address, reply-to fields, paths received, and timestamps to work backward and determine the true origin of the email.

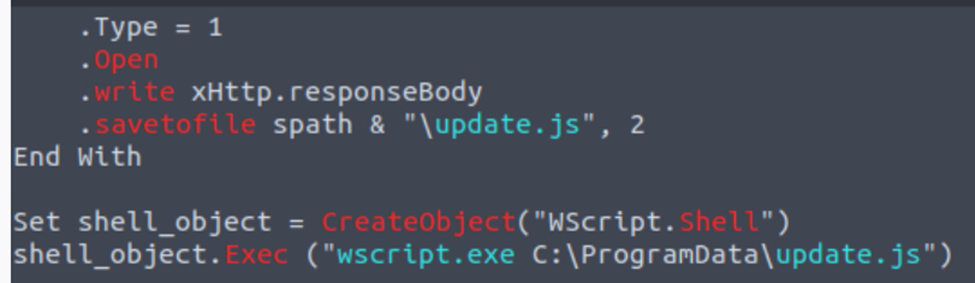
We also downloaded the malicious attachment, a Word document named Resume.doc, to our Ubuntu VM. In the terminal, we ran the olevba command followed by the resume file

*olevba Resume\_WesleyTaylor.doc*





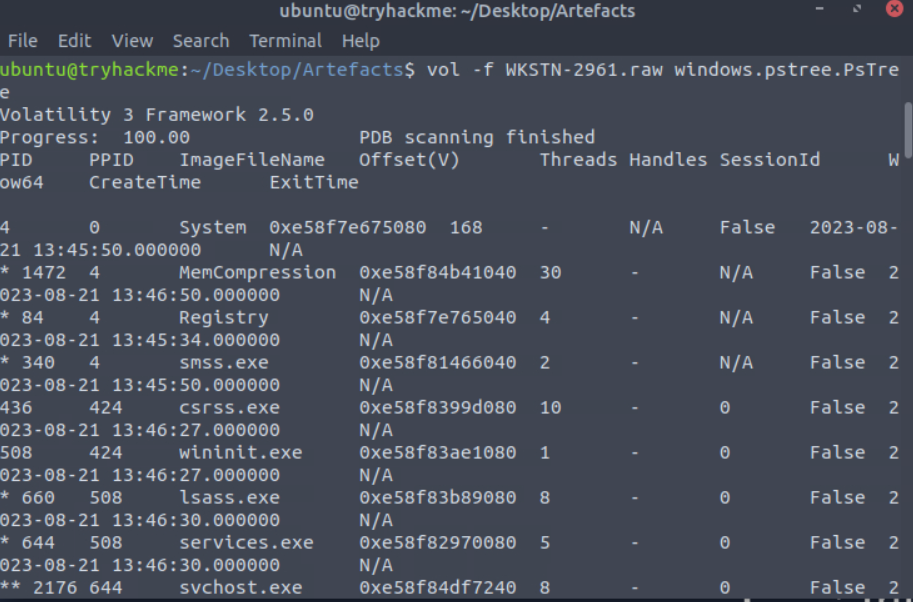
to extract all VBA macro codes from the attached resume and display all suspicious programming keywords that are commonly used in malware. We learned that the VBA scripts embedded in the resume were designed to download a JavaScript payload update.js from an external attacker-controlled server.



To understand the attacker’s activity after initial compromise, we captured a memory dump of the infected virtual machine while it was still running. We then used Volatility to analyze the dump and uncover suspicious behavior.

We ran the following volatility command on the memory dump file:

*vol -f WKSTN-2961.raw windows.pstree.PsTree*



Which revealed the presence of wscript.exe—a known LOLBin used to execute malicious scripts. We learned that wscript.exe had executed the update.js script.

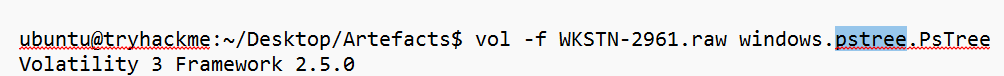
In addition, we extracted the system logs to check for any outbound communications enabled by the VBA scripts or any additional payloads that might have been downloaded by the JavaScript payload. We also looked at any scheduled tasks or startup items that had been created after the payload had been downloaded to see how the attacker maintained persistent control of the compromised employee’s computer.

**Artefacts Analysis:**

We first analyzed the attached Word document in the phishing email(Resume\_WesleyTaylor.doc) using the Olevba tool on our Ubuntu VM. The tool extracted the embedded VBA macro code and flagged several suspicious keywords: AutoOpen, Shell, and CreateObject. These are well-known programming keywords that are known to be malicious. AutoOpen allowed the macro to execute automatically when opening the document. CreateObject and Shell enabled the script to interact with Windows’ native script-executing file (wscript.exe). The VBA macros embedded in the resume connected the client to a remote malicious server and downloaded a JavaScript file named update.js. The JavaScript file was then executed by wscript.exe from the client’s native environment. This approach bypassed the Windows Defender Firewall.

To validate this execution path and gain further insight into post-exploitation activity, we analyzed a memory dump (WKSTN-2961.raw) of the infected virtual machine using Volatility.

The windows.pstree plugin revealed that wscript.exe had been launched as a child process of WINWORD.EXE, confirming that the macro executed a secondary payload as designed.







To further analyze execution context, we used the cmdline plugin, which showed that wscript.exe had been executed with a command pointing to update.js, validating the downloading of update.js once a user opens the resume template.

For identifying the PID of the process that executed the stage 2 payload we used the command : *vol -f WKSTN-2961.raw windows.pstree | grep wscript.exe*. The PID was *4260*

For identifying the URL used to download the malicious binary executed by the stage 2 payload we used the command : *strings WKSTN-2961.raw | grep boogeyman*. The URL was

*https://files.boogeymanisback.lol/aa2a9c53cbb80416d3b47d85538d9971/update.exe*

The PID of the malicious process used to establish the C2 connection was identified by using the command : *vol -f WKSTN-2961.raw windows.pstree.PsTree | grep 4260*.

From the process list we can see wscript.exe has a child process named updater.exe so this must be the process establishing c2 connection.

The full file path of the malicious process used to establish the C2 connection was identified using the command: *vol -f WKSTN-2961.raw windows.cmdline | grep updater.exe*

This showed the command line used to start the process, including the full path of the executable: *C:\Windows\Tasks\updater.exe*

The IP address and port of the C2 connection initiated by the malicious binary file was discovered using the command: *vol -f WKSTN-2961.raw windows.netscan | grep updater.exe*. The result was *128.199.95.189:8080*

Furthermore, we looked into possible ways to sustain access to the system. We studied the system logs. These told us that a scheduled task had been created soon after the macro had run. This task was responsible for starting up stage2.ps1, a PowerShell script that was part of a follow-up payload that our JavaScript had delivered. We concluded that the zip file granted the attacker’s persistent control of the compromised client machine.

Furthermore, logs indicated that there was outbound traffic that indicated that the client was in communication with a malicious server. This mechanism could allow the attacker to potentially download further payloads and exfiltrate data, although in this scenario, there was no data exfiltration that was being simulated.

**Document Findings:**

The attack started with a simulated phishing email sent to Quick Logistics LLC's HR sector. This email pretended to be from a job applicant and included a nasty piece of work pretending to be a resume from a guy named Wesley Taylor. The HR employee, Office 365 Outlook 2019, opened this file. Once it was opened, the malicious code hidden within the VBA (Visual Basic for Applications) macros was executed automatically, thanks to a neat little function called AutoOpen.

The sequence of commands in the macros used CreateObject and Shell, letting them take advantage of wscript.exe, a valid Windows scripting utility (LOLBin). With this setup, they quietly downloaded and executed a second-stage payload, update.js, from an outside server—an attacker-controlled one. They accomplished all this while bypassing traditional security defenses and minimizing red flags.

Once update.js was run, the infected computer reached out to a remote Command and Control (C2) server. This script was registered as a scheduled task on the system, which means it is allowed to automatically execute at system startup and thus be persistent.

Looking at the memory dump (WKSTN-2961.raw) through the lens of Volatility, we saw a suspicious process tree. The windows.pstree plugin indicated that wscript.exe was spawned as a child of WINWORD.EXE, which meant it was directly linked to the opened resume. The cmdline plugin confirmed that wscript.exe was used to execute update.js.

Key Indicators of Compromise (IoCs) included:

1. The file Resume\_WesleyTaylor.doc contains harmful macros.
2. Suspicious usage of VBA keywords: AutoOpen, Shell, CreateObject
3. WINWORD.EXE running wscript.exe
4. Outbound links to a distant C2 server
5. Task Scheduler task that runs stage2.ps1.

We will include the data results of the execution of the volatility command made for this analysis in the file VolatilityData.txt..

**Incident Response**

The incident response playbook outlines a comprehensive approach to managing phishing-based cyber threats, specifically targeting HR staff with macro-enabled documents.

The phase of preparation highlights taking preemptive security steps. It urges a security posture that moves from a reactive to a proactive state. This transitional state ensures that not just incident response teams but all units of an organization are prepared for a security incident. This level of preparedness arises from conducting regular training for both awareness and the specifics of a planned incident response. In the preparation stage, the meeting of preventive repair with the maximum effort of incident response serves to equalize the intelligence tower during the initial resolve phase.

During detection and analysis, analysts are directed to search for anomalies like email metadata mismatches or Office apps that spawn scripting engines (e.g., WINWORD.exe launching wscript.exe). Security teams should use SIEM or EDR solutions to find abnormal behaviors in PowerShell or JavaScript and closely examine artifacts such as memory dumps and system logs.

To contain the situation, it is essential to promptly isolate the infected system from the network. Following that, memory and disk captures should be taken to keep the forensic evidence intact. Analysts should retain any and all emails and documents that even look suspicious for a more detailed examination later. And they should make sure that any user accounts tied to the infected system are secure—probably by changing the passwords, and a good case can also be made for temporarily deactivating any accounts, just to be safe.

Eradication efforts consist of the removal of harmful files such as Resume\_WesleyTaylor.doc, update.js These efforts include scrubbing registry changes and removing startup or scheduled task entries that allow the malicious code to run at boot time. All this should be done in conjunction with using antivirus software and security patches to completely neutralize any outstanding vulnerabilities.

During the recovery phase, it is paramount to install valid, clean backups and restore system and network monitoring to ensure that any reinfection will be caught immediately. Only after confirming system integrity should normal operations resume.

Finally, the steps taken after an incident involve debriefing the team, updating the training materials, and ensuring the indicators of compromise (IOCs) gathered during the incident are fed into the organization's threat intelligence system. Conducting debriefing sessions with those involved in contained incident operations is critical to organizational learning and future incident readiness.

**Evaluation of IR with NIST Framework**

*Preparation:*

Before the simulated incident, Quick Logistics LLC had made outstanding progress in its preparations. All endpoints had Endpoint Detection and Response (EDR) software installed, and the email systems used SPF and DKIM to verify the legitimacy of senders. Moreover, employees got regular security awareness training. Nonetheless, there were significant gaps. There was no inventory of assets or classification of them by criticality. There were no baseline profiles for either typical email traffic or user behavior, which made anomaly detection extremely difficult.

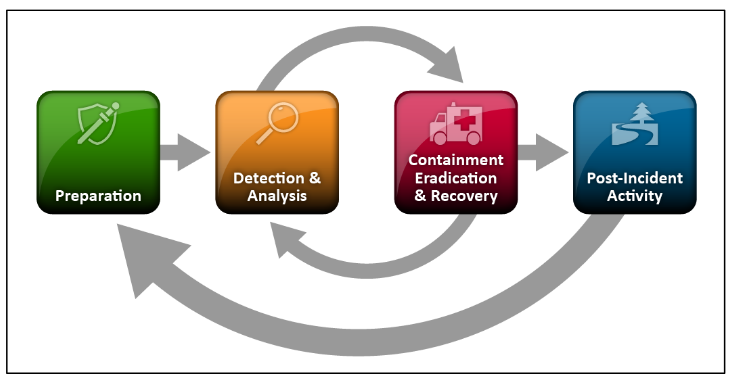
*Detection and Analysis:*

The attack involved sending an email with a dangerous attachment—one that had a filename like the ones you'd expect for a job application. Included with the attachment was a malicious script that was enabled by a common Office application feature called "AutoOpen macros." This feature essentially lets the application run some code when you first open a file. In this case, the code that ran was very bad news for anyone whose computer or network happened to be in the path of the malware.

*Containment, Eradication, and Recovery:*

Once detection took place, impacted systems were cut off to stop any additional spread. Preserved memory dumps followed standard forensic procedures. Ridding systems of malware involved deleting bad scripts, cleaning registry keys, and reviewing any suspicious scheduled tasks that remained. The process was dogged by two things, though: a lack of asset criticality designations and the absence of defined recovery time objectives (RTOs). Because we couldn't prioritize effectively, it took much longer than it should have to get things back to a semblance of normal, even though the compromised VM was successfully reset and hardened afterward. Post-Incident Activity:

The collected indicators of compromise included the use of AutoOpen, the invocation of wscript.exe, a scheduled task that executed stage2.ps1, and evidence that we had made outbound connections to a C2 server. These findings resulted in some playbook changes. We documented the lessons we learned, but there was no formal review involving HR, Legal, or Communications. We didn't establish or follow any metrics like MTTD or MTTR.



**Lessons Learned**

This simulation highlighted some vital aspects of where the defensive position can be strengthened. It showed that, when it comes to the initial attack vector, phishing is still way out in front, and even trained security employees are not immune to being deceived by some very convincingly crafted bait. The bait may be even more convincing since HR deals with so many external communications. That makes interactive, role-specific phishing simulations all the more crucial.

Documents that use macros can also be a constant source of threats. Hence, enforcing strict group policies in Office to block all macros in documents from untrusted sources is essential.

Also, attackers often take advantage of Living-off-the-Land Binaries (LOLBins) such as wscript.exe and powershell.exe to masquerade as legitimate processes. These tools should be monitored very carefully, or restricted if possible.

The examination underscored the vital role that sound logging and forensic preparedness play. When memory is captured and logs are kept in a detailed manner, the attack's complete sequence can be reconstructed.

Persistence is often achieved by attackers via Scheduled Tasks or startup scripts. These might seem low-tech compared to other infiltration methods but are nonetheless effective ways to maintain access and control. Unfortunately, they are also low-hanging fruit for the defensive team. Accordingly, defenses should monitor both of these methods closely.Preparedness is critical: A current incident response playbook with well-defined roles and responsibilities is a force multiplier for the efficient and well-coordinated response to an incident. All identified indicators of compromise (IOCs), including things like file hashes and C2 domains, should be documented and shared both within the organization and with trusted partners to support a more expansive and effective effort at threat intelligence.

**Conclusion:**

This project offered the chance to simulate a phishing-based cyberattack, like the ones that often hit businesses in the real world. Most safe environments have some basic defenses (like firewalls) that the bad guys first try to bypass. We used a phishing email as our way in, and to optimize the attack we carefully constructed the malicious Word document that the email urged the target to open. Inside this document, we used several tricks—including the old favorite of hiding a malicious macro behind a harmless-looking one—that are often seen in real-world breaches.

Completing this simulation has made us better at identifying indicators of compromise and tracing the tactics used by attackers. We've also been reminded of the essential value of logging, timely memory capture, and detailed process analysis in doing incident response right. And we've certainly been given a refresher on the importance of continuous awareness training, effective macro policy enforcement, and having a well-defined response playbook to contain and recover from these sorts of threats.