(Threat Hunting) Kill Chain 1: "Go Phish: Visualizing Basic Malice"

Come take a dive into the data lake and cast some queries to find proof that users have run files from malicious actors. How can we prove the existence of troublesome activity in the environment? We will take a journey as if we are a new member of the Magnum Tempus Financial Security Team and proceed through a Threat Hunt through the eyes of a newbie in the field of Threat Hunting.

Overview

What will we learn?

There are a number of concepts we will go over and learn in this walkthrough:

- What is phishing and what is a phishing payload?
- What is Visual Basic for Applications?
- What are Macros and what does this have to do with phishing and Threat Hunting?
- How can we walk through the thought process associated with a Threat Hunt from hypothesis to tangible results?
- What tools do we have at our disposal and what can we do with them?
- Can we go deeper and find out more after validating our hypothesis?

Initial Required Concepts

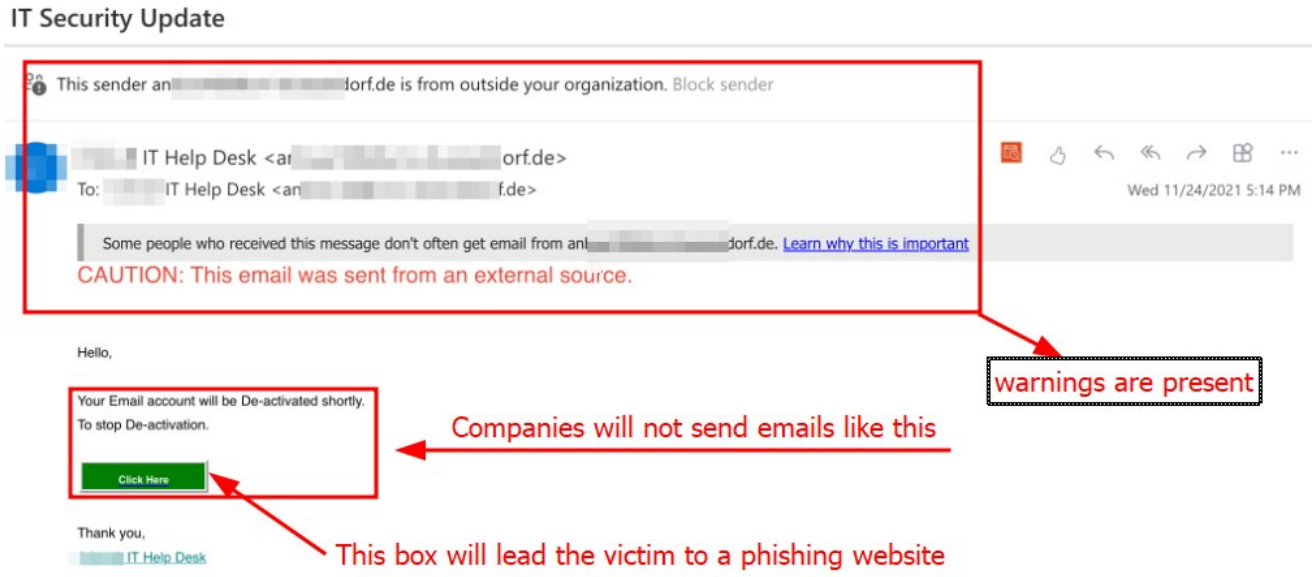
In order to dive into the hunt, we need to have some baseline information to better understand what we are seeking to find. Let us take a look at some of these now.

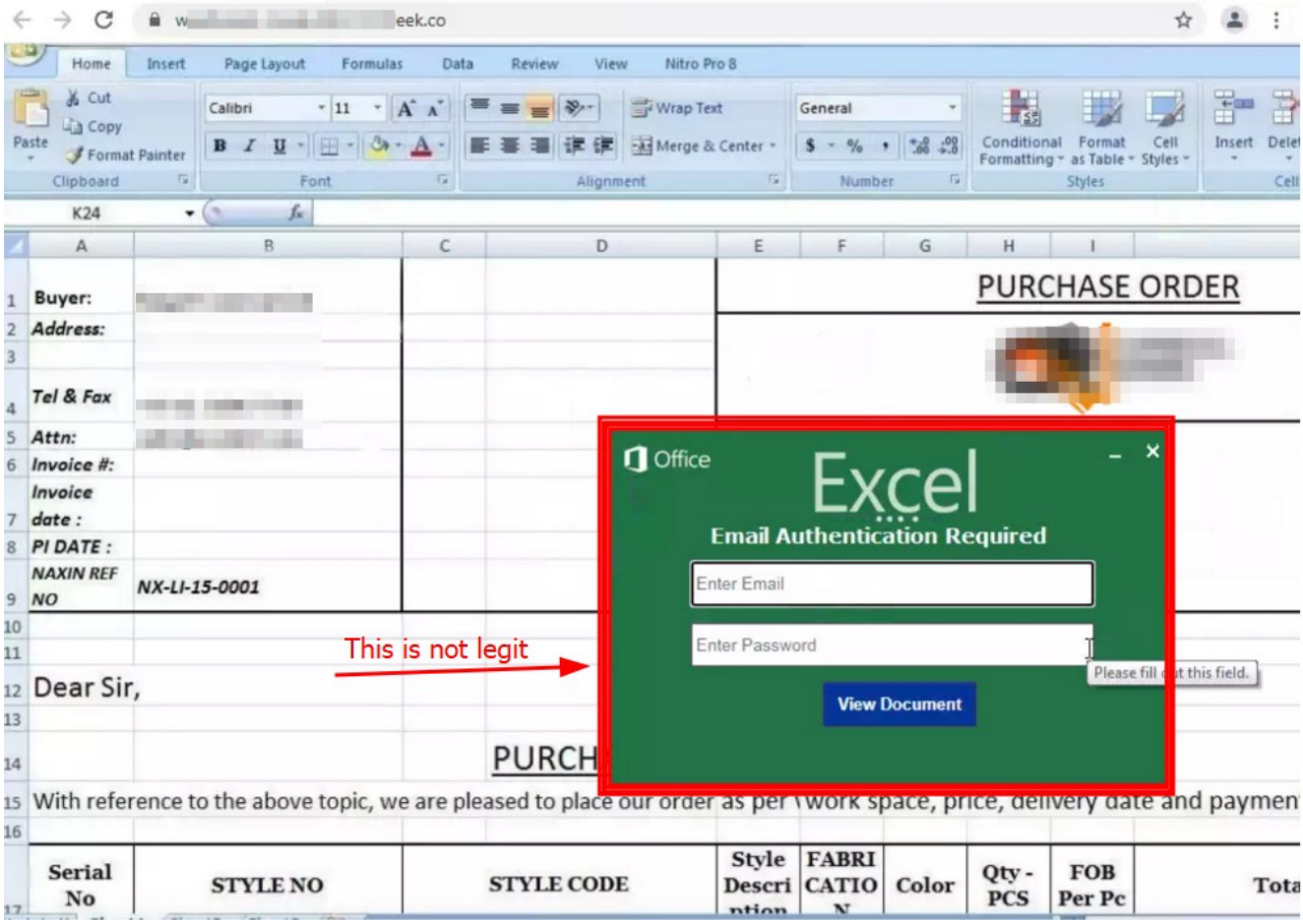
What is phishing?

In simple terms, phishing in this context is an attempt by an adversary to use methods to obtain credentials in an illicit manner using email communications.

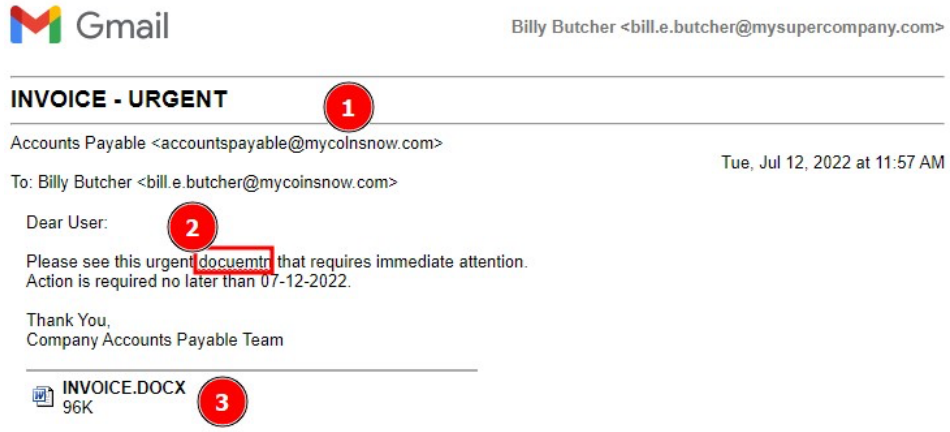
What is a phishing payload?

A phishing payload is the means in which an adversary attempts to obtain the credentials of a target. The payload could be a link that sends an unsuspecting victim to a webpage that emulates a known website (like Microsoft Office 365 portal or Banking logon page) to collect usernames and passwords. Here are a few examples of malicious emails I have seen in my information security experiences:





Another type of payload can be a document or file specially crafted to behave in a certain manner or take specific actions to attempt to collect credentials from a user's system:



Let's look at what we see in the above screenshot:

1 . The company email is mycoinsnow.com but the email sender is mycolnsnow.com - this is a typosquat of the domain.
2. Misspelling 'document' in the text. Additionally, creating a sense of urgency on the recipient that action is required by a specified date.
. Document attached is a .docx file that simply says "invoice" with no other indication on what it is for.
3

Attacks using the above type of email often utilize the Visual Basic for Applications (VBA) coding present in Microsoft Office Suite, primarily in Microsoft Word, Excel, and PowerPoint.

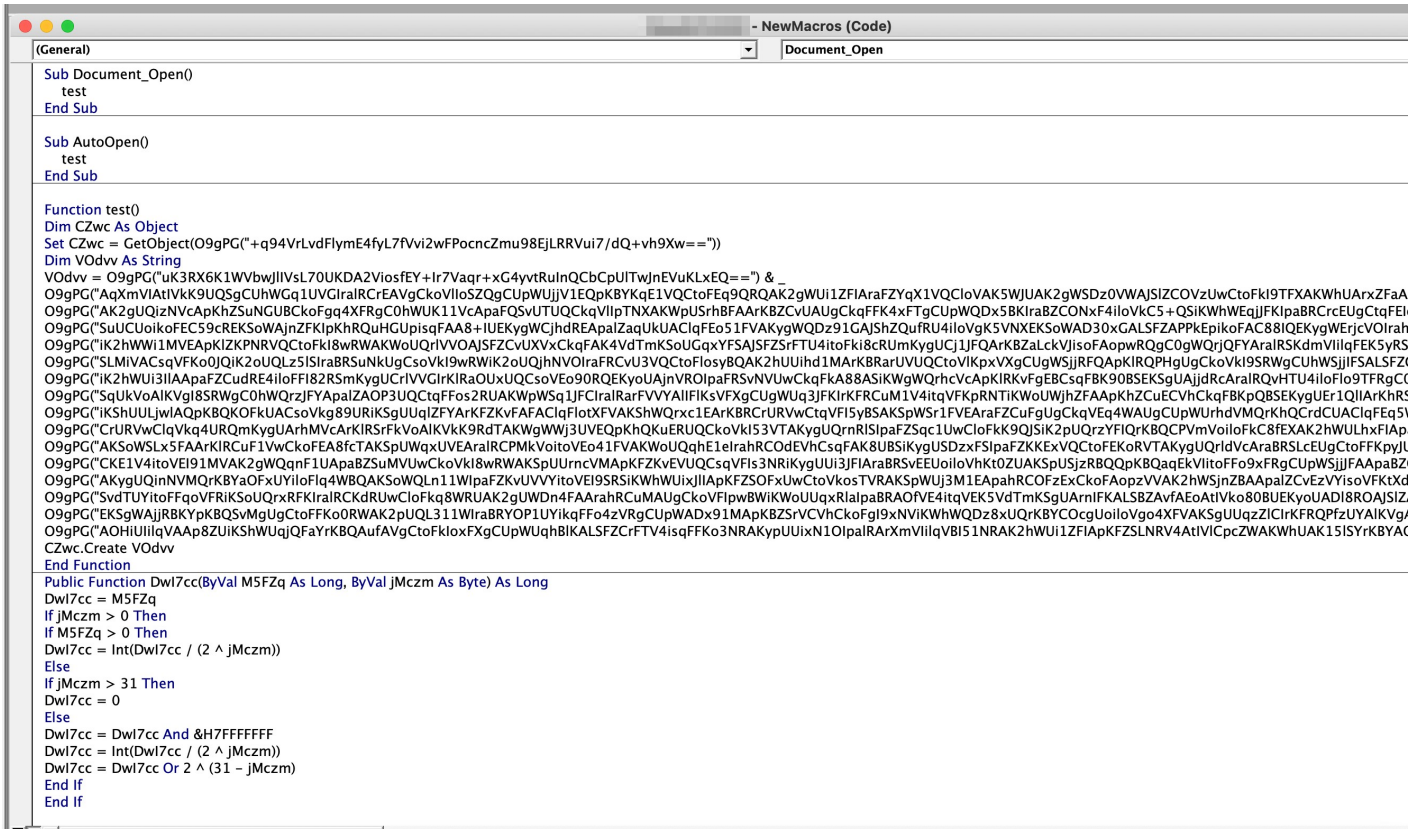
What is Visual Basic for Applications?

Visual Basic for Applications (VBA) is a slightly stripped down/limited version of Visual Basic coding language that can be used in Microsoft applications to add functionality and extensibility not natively available in the applications themselves. This can be useful in helping automate repetitive actions or automate retrieval of data when certain options are chosen in a document. VBA allows system-level access to perform actions and can act outside of the Microsoft Office Applications as well. We commonly call these "Macros".

What are the issues with Macros?

Due to how the functionality works in allowing access outside of the walled garden of Microsoft Office Suite, macros can allow an adversary the ability to make system changes or download additional malicious code via these macros with little interaction from the victim.

An example of how the VBA Macro document could appear in a Microsoft Word document can be see here:



Now that we understand what phishing, phishing payloads, VBA, and macros are, let's put these concepts together to see if we can build a Threat Hunt with these ideas in mind!

Threat Hunting

What is Threat Hunting?

Threat Hunting (TH) is a process of being proactive of unveiling unknown-knowns and unknown-unknowns to better our security posture **Hypothesis**

In order to begin the Threat Hunt, we need to have a reason to start the hunt. To start, we will need to come up with a hypothesis.

What is a hypothesis?

A hypothesis can be described as something we think might be occurring or something that we think might be taking place in an environment. A hypothesis focuses on the 6 W's:

- Who: Who is doing the activity?
- What: What is happening?
- Where: Where (What systems/networks) is this happening?
- When: What time/time period did this happen?
- Why: What is the end goal for the activity being performed? How:
- How is the activity occurring in the system?

How do we create a hypothesis?

For HOW to create a hypothesis for Threat Hunting, you can read an in-depth guide here: (insert link).

For now we will go over the hypothesis I have created for this scenario.

Let's create a hypothesis to hunt for potential phishing activity!

Broad Hypothesis

Magnum Tempus employees receive malicious phishing emails.

Narrow hypothesis

MT Employees are targeted with Malicious Microsoft Documents containing VBA macros via phishing email. Some employees will click and open malicious documents.

Transitioning from a Hypothesis to a Query

How can we formulate a query based on our hypothesis?

Our initial hypothesis presumes Magnum Tempus employees receive emails with malicious documents and opens those documents.

First we need to understand what tools we have available. For this Hunt, we are using Splunk, however many of the procedures/methods we use can and should be used with other query languages/log platforms.

Since we know our log data is in Splunk, we first need to determine what log sources we have available to us.

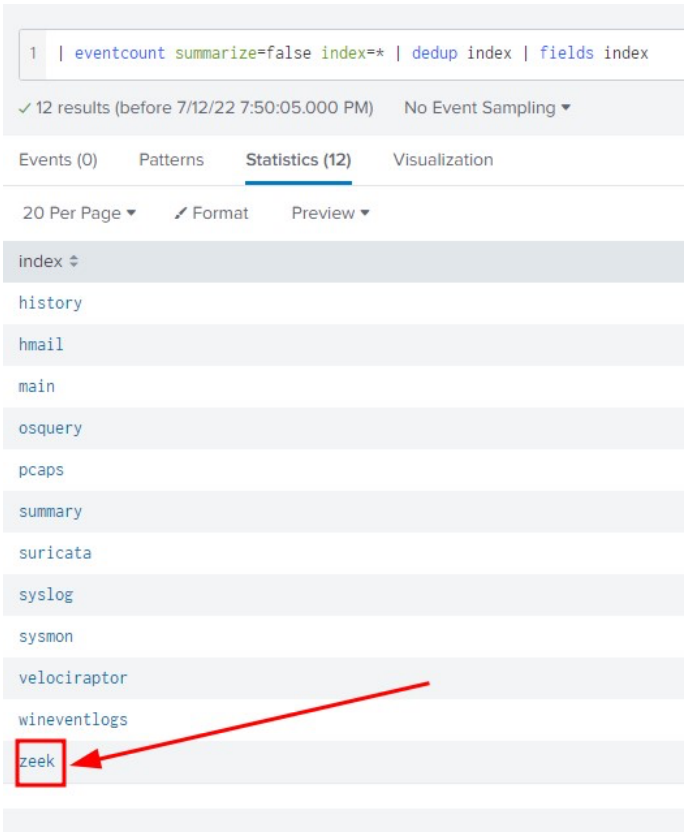
We could modify our index to include ALL log source data using a wildcard query like this:

```
index="*"
```

HOWEVER, this is considered bad practice as doing so is an expensive (resource-intensive) query and could cause undesirable effects on the Splunk (or other Log Aggregator) server and could hinder other searches being done on the system.

Thanks to our friend, Cereal Killer, we can utilize the following query to find the indexes (source) of data to query against:

```
| eventcount summarize=false index=* | dedup index | fields index
```

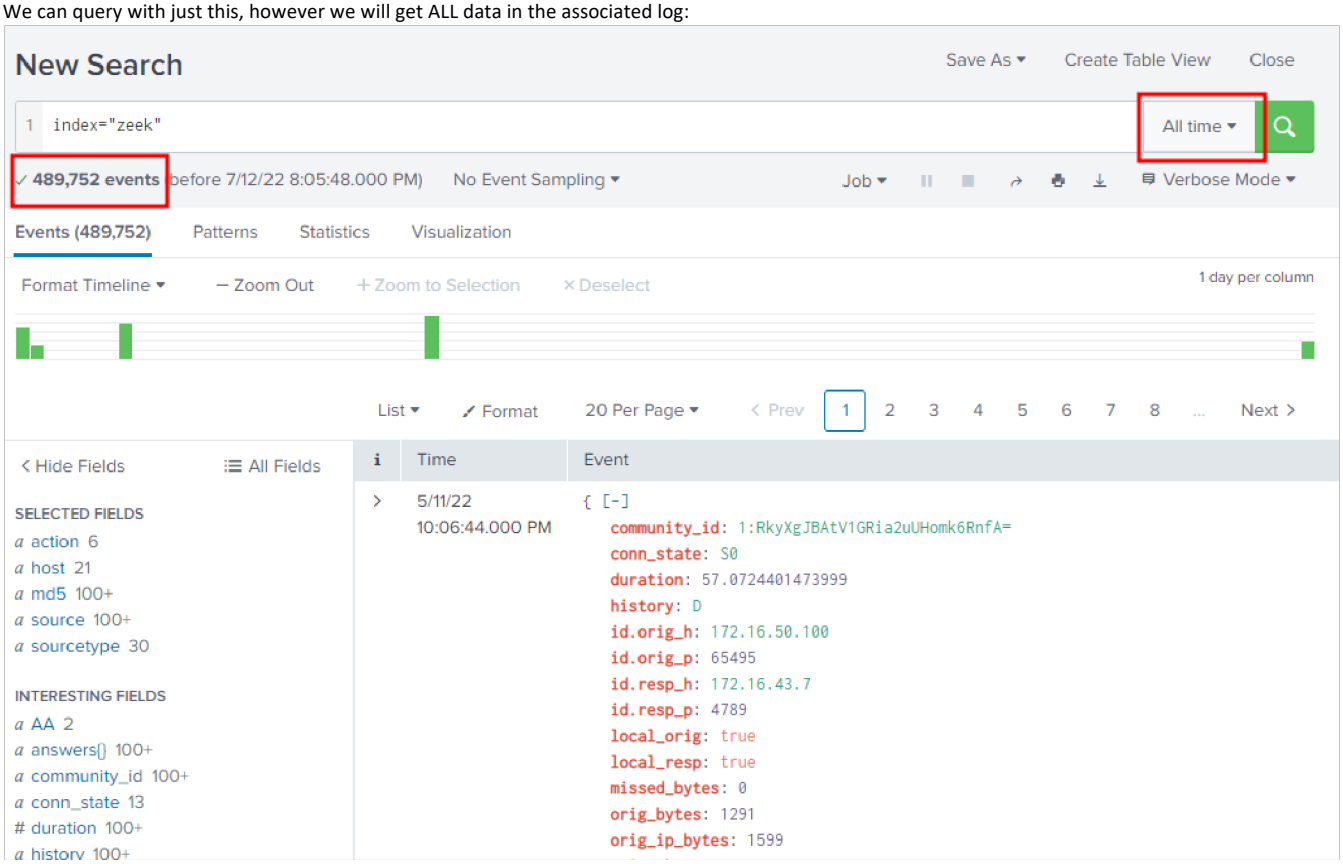


We will start with Zeek logs in Splunk because we can find documents transmitted over network traffic if they are downloaded from a company email on a company asset. Zeek is used to collect telemetry data and traffic flowing through an organization's network.

In Splunk queries, we start by calling the index (source) of the data we are querying against:

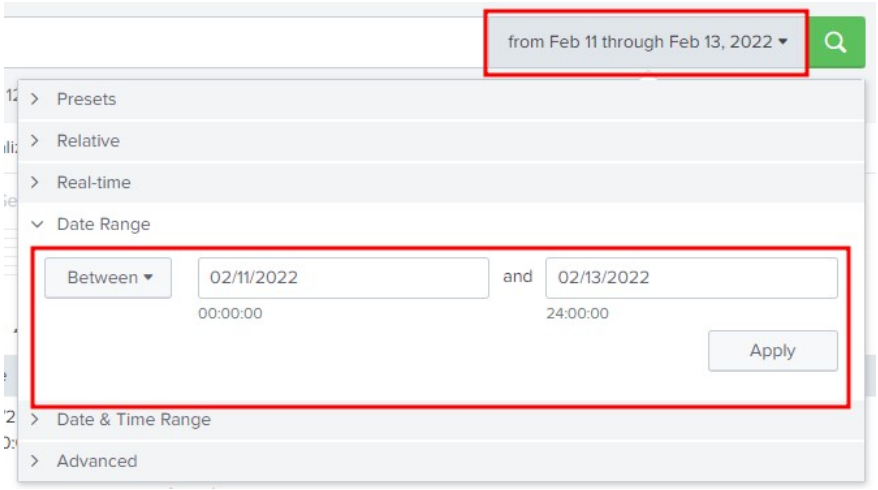
```
index="zeek"
```

We can query with just this, however we will get ALL data in the associated log:



> NOTE:
> We need to change the time filter to the time period we are checking against for this specific scenario because we don't know exactly when the log data starts/stops.
> In your own Threat Hunting (outside of this scenario), you will want to identify a timeframe to conduct your searches. This could be 24 Hours, 1 Week, 1 Month.

In this scenario for Magnum Tempus, we are focusing on 2022-02-11 through 2022-02-13:

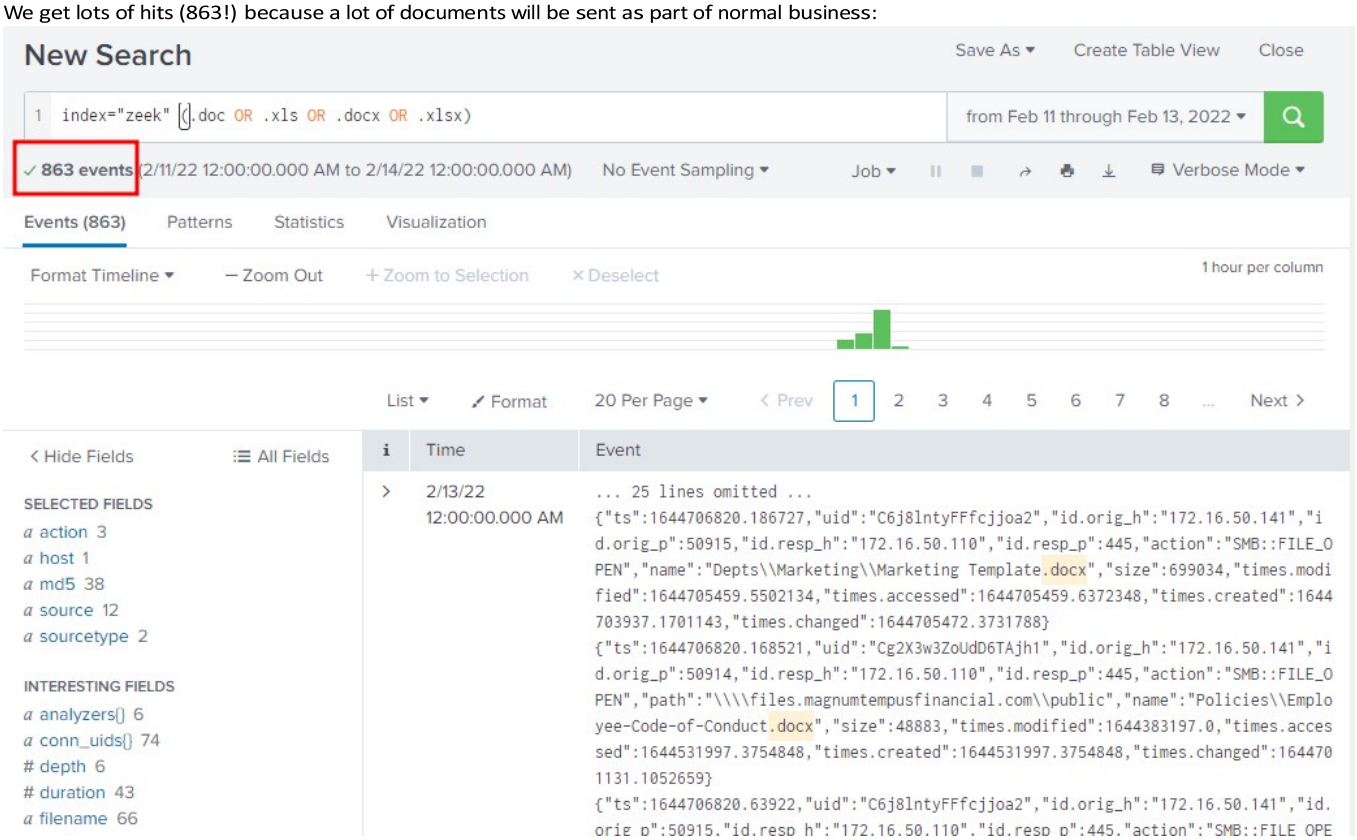


Let's focus on document file types that are often sent as potential phishing attachments.

Some of the more common formats include Microsoft Word formats (.doc, .docx) and Microsoft Excel Spreadsheets (.xls, .xlsx). Splunk allows us to query for multiple file types at once. Note we are not necessarily looking for an official filetype designation because we may not know exactly if this is indexed in Splunk. Let's check for the filetypes specifically as shown here:

```
index="zeek" (.doc OR .xls OR .docx OR .xlsx)
```

We get lots of hits (863!) because a lot of documents will be sent as part of normal business:



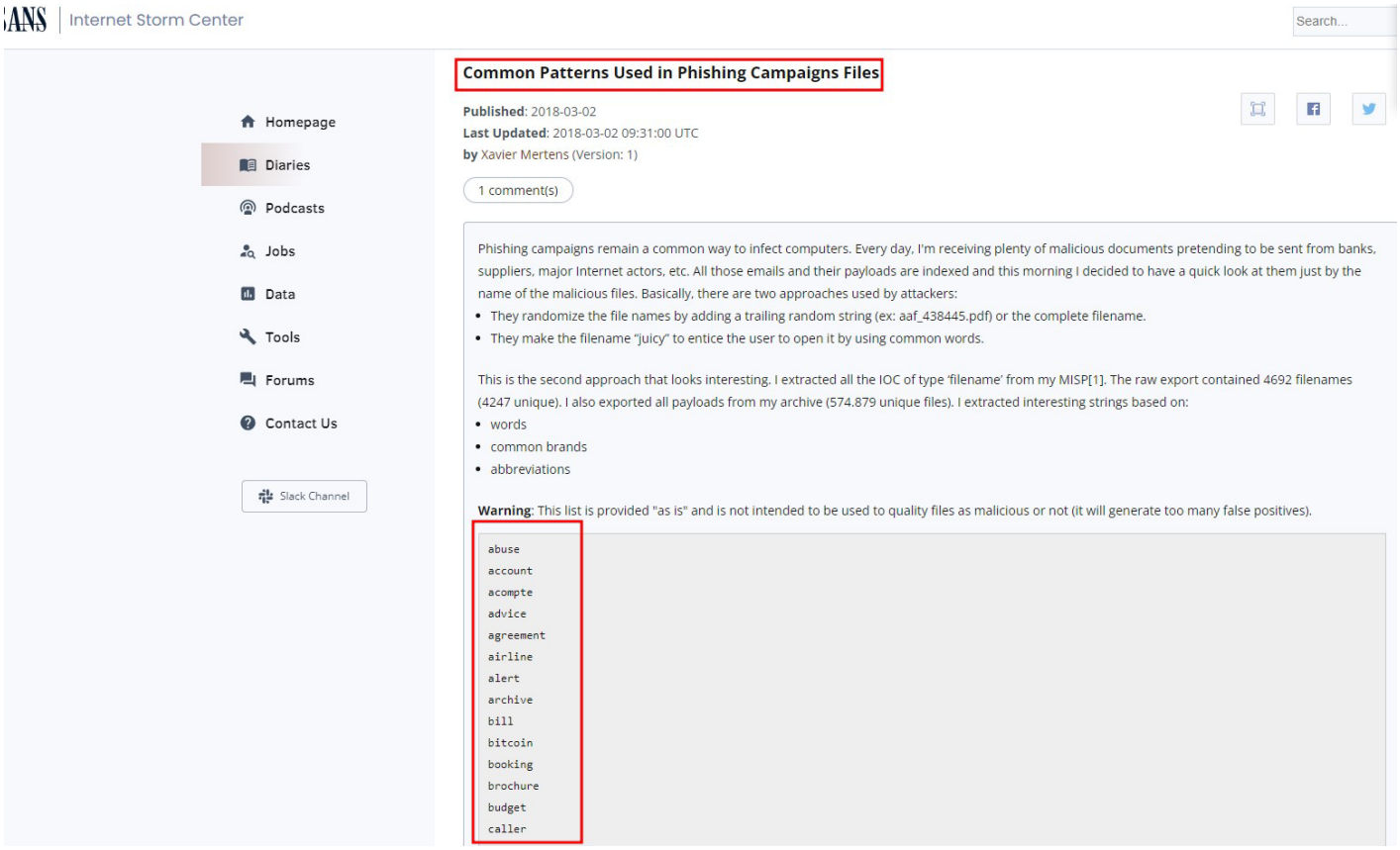
Let’s try to pare down the total number to a more manageable number using some common phishing terms.

How can we refine this query?

Let's start with the previous query:

```
index="zeek" (.doc OR .xls OR .docx OR .xlsx)
```

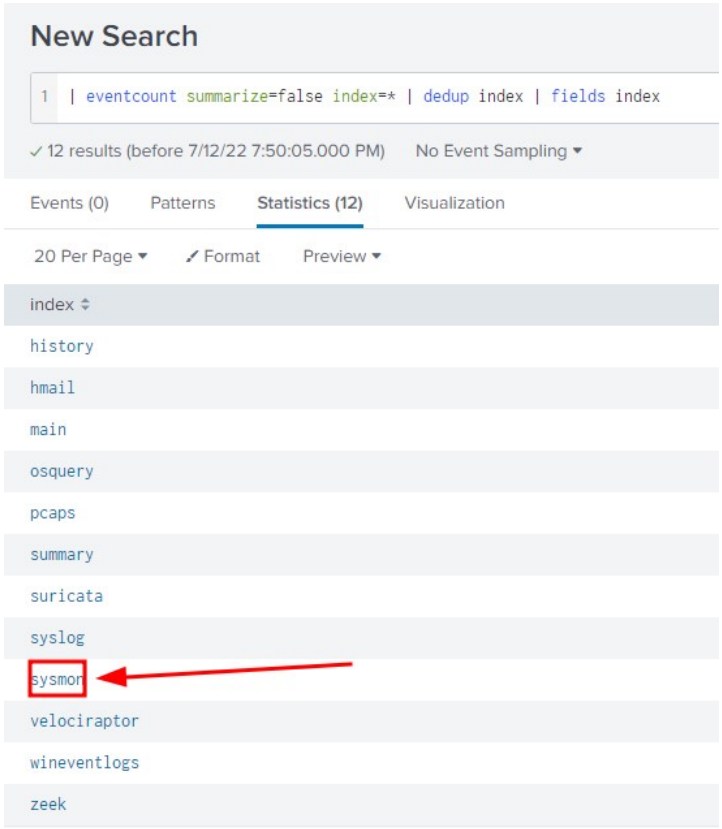
This gave us way too many results. We want to reduce the number of files, but need to find files that could indicate potential phishing activity.
Occasionally we see common terms/names in documents related to phishing activity as seen here at SANS Common Patterns Used in Phishing Campaign Files:



Let's pick a few and run another search, this time expanding our search to other log sources within Splunk (using index IN):

Let's try including a different log source with our zeek query.

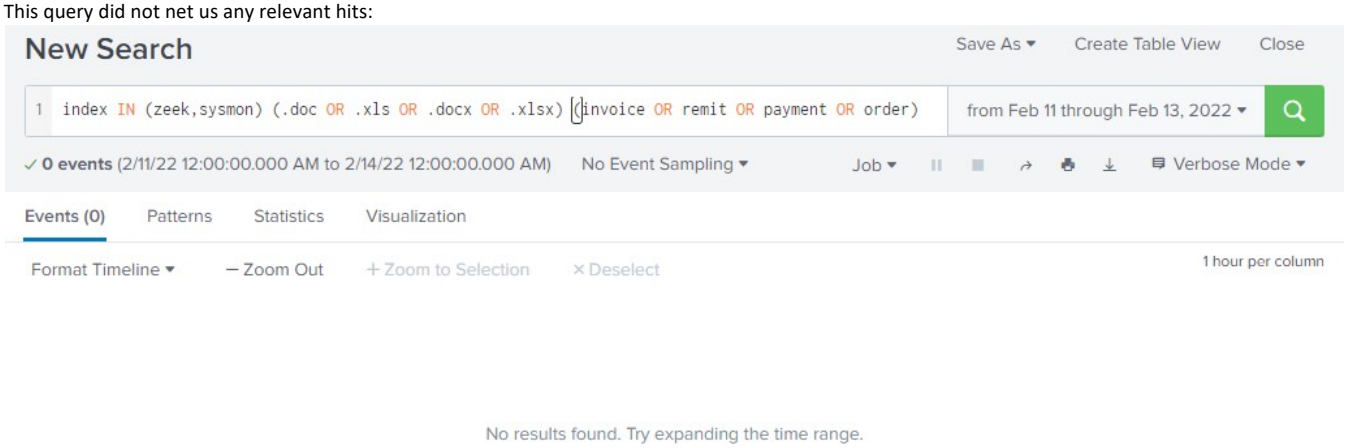
Looking at our index from before, let's go with sysmon, as we might be able to see files accessed on endpoints (Much like how Zeek is used for network telemetry, Sysmon is used to generate telemetry data from endpoint devices in an organization.)



We can modify our search as seen here:

```
index IN (zeek,sysmon) (.doc OR .xls OR .docx OR .xlsx) (invoice OR remit OR payment OR order)
```

This query did not net us any relevant hits:



We should probably try a different approach.

How can we refine this query since the previous did not get us what we expected?

Let's start with the initial query:

```
index="zeek" (.doc OR .xls OR .docx OR .xlsx)
```

Our initial hypothesis presumes that users executed malicious VBA macro code. Is there a way for us to determine based on log data that such a document was executed?

In a word: yes.

Reviewing SANS Office macro execution evidence we see that we can check logs for “TrustRecords” to see if there were Windows Registry modifications:

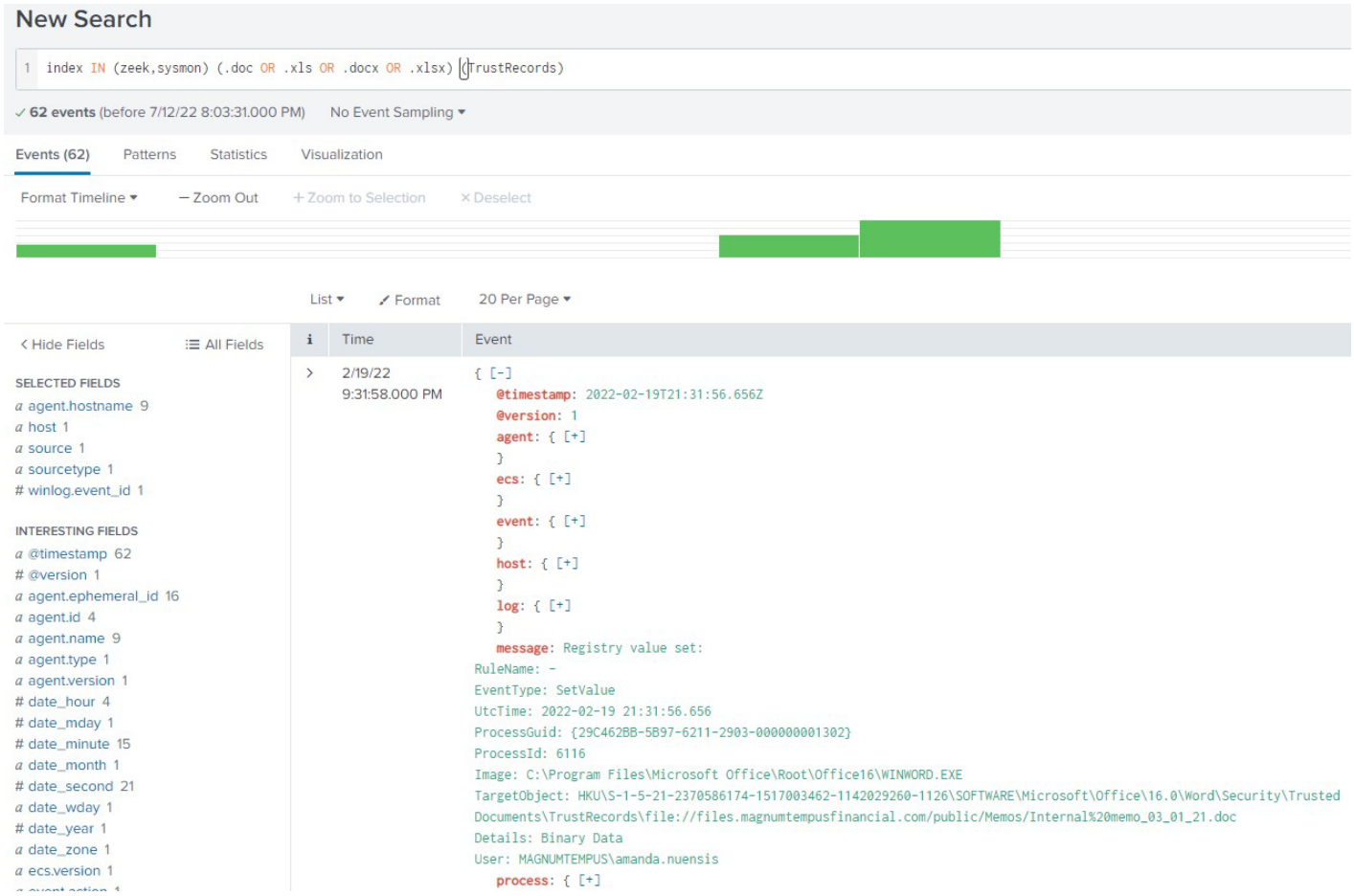
> One of the few places where macro execution leaves traces is in the "TrustRecords" entry in the registry:
> HKCU:\SOFTWARE\Microsoft\Office\16.0\Word\Security\Trusted Documents\TrustRecords
> HKCU:\SOFTWARE\Microsoft\Office\16.0\Excel\Security\Trusted Documents\TrustRecords
> HKCU:\SOFTWARE\Microsoft\Office\16.0\PowerPoint\Security\Trusted Documents\TrustRecords
>
> -From https://isc.sans.edu/diary/Office+macro+execution+evidence/27244

Let’s take a look at what we should modify our search query to:

```
index IN (zeek,sysmon) (.doc OR .xls OR .docx OR .xlsx) (TrustRecords)
```

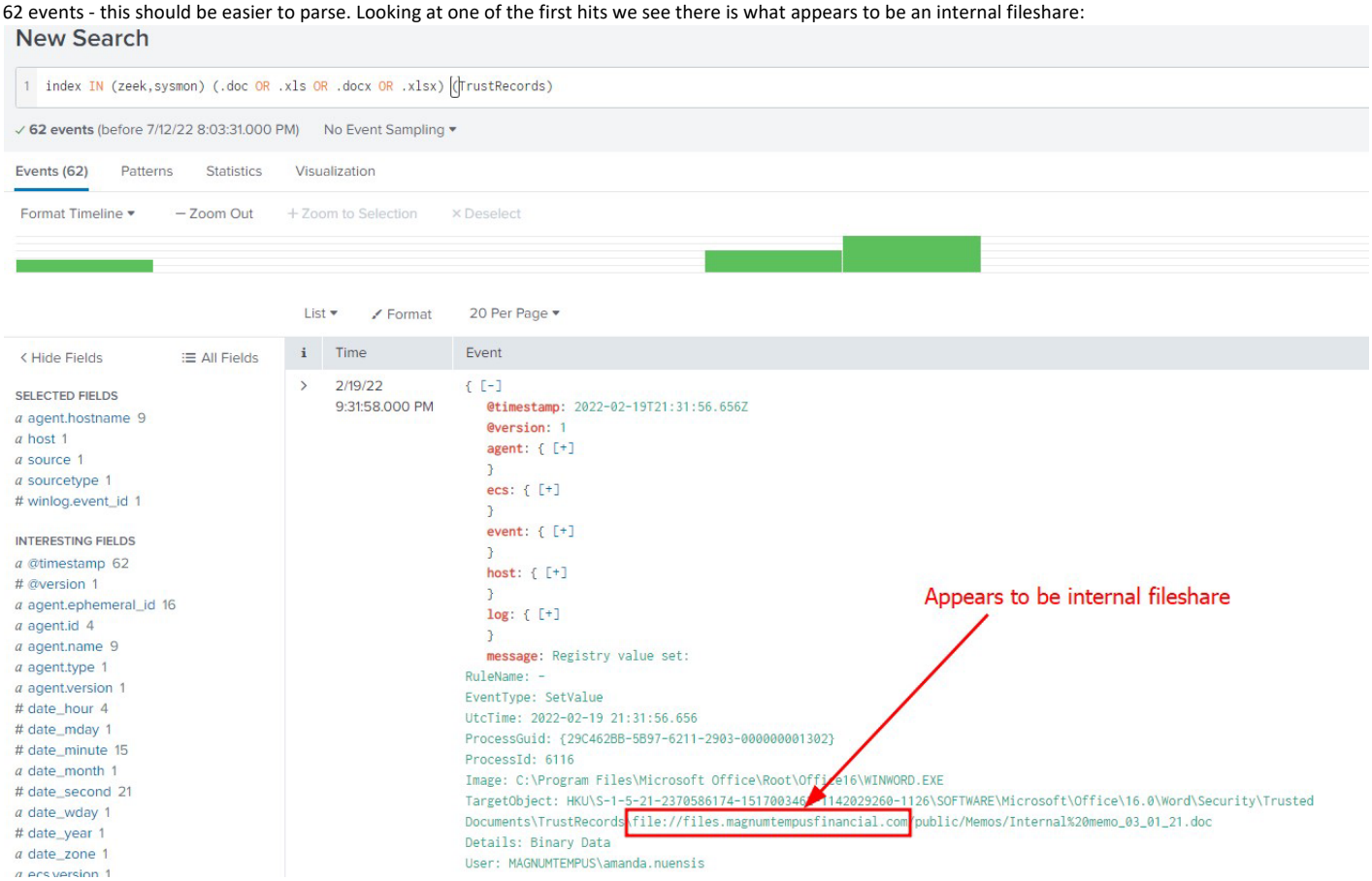
Let's run this query!

We have hits!



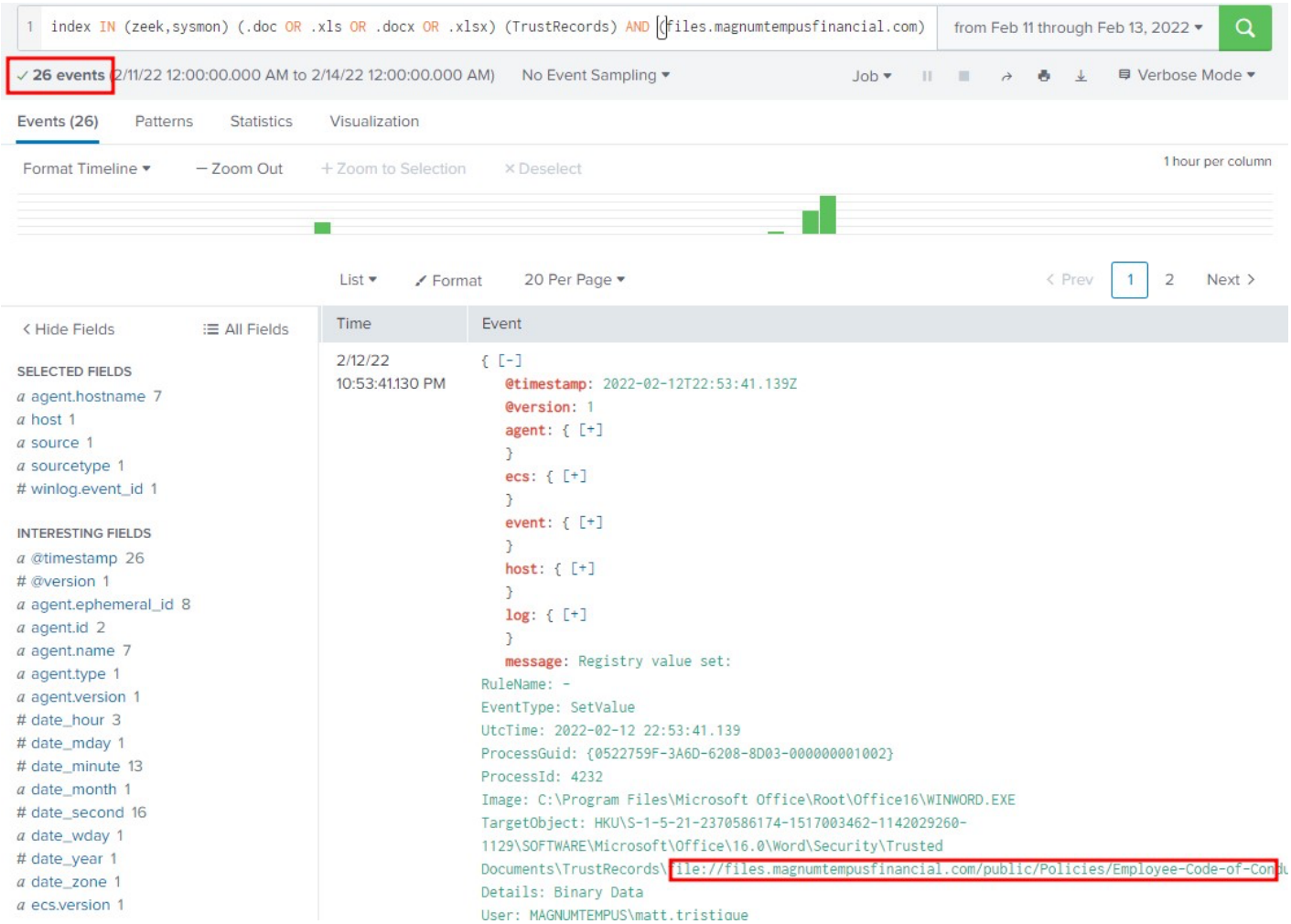
Let's look at the overall results:

62 events - this should be easier to parse. Looking at one of the first hits we see there is what appears to be an internal fileshare:

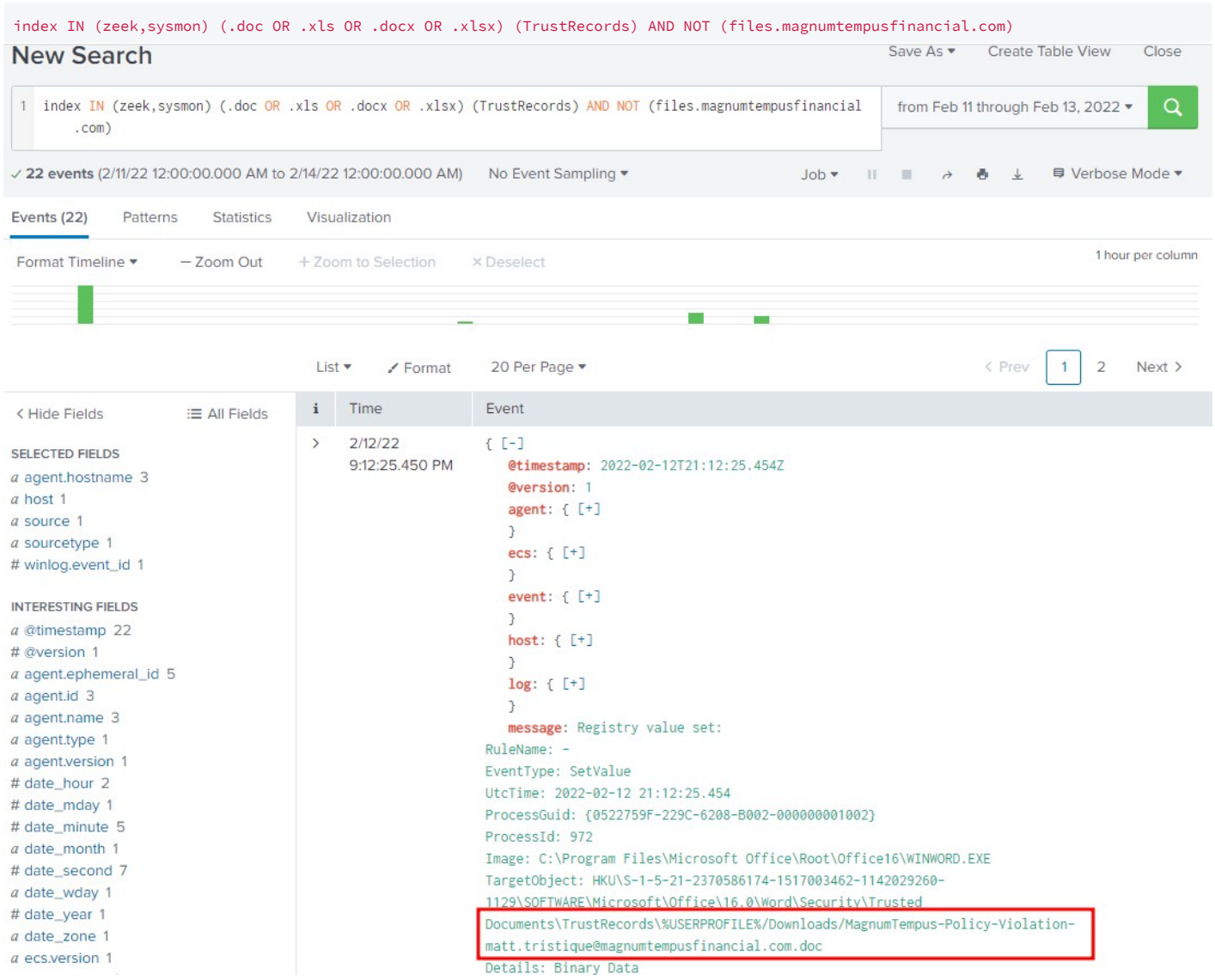


```
files.magnumtempusfinancial.com
```

For purposes of this TH, let's exclude this in the query. Since our initial hypothesis indicated that users would download and execute malicious documents downloaded from a malicious sender, we might not expect to see these files in an internal file share at first. It is possible, however, that files could be malicious and saved to the fileshare. Let's take a look to see how many of our documents that use macros are on the internal fileshare: `index IN (zeek,sysmon) (.doc OR .xls OR .docx OR .xlsx) (TrustRecords) AND (files.magnumtempusfinancial.com)` Running this, we see 26 events that match this query:

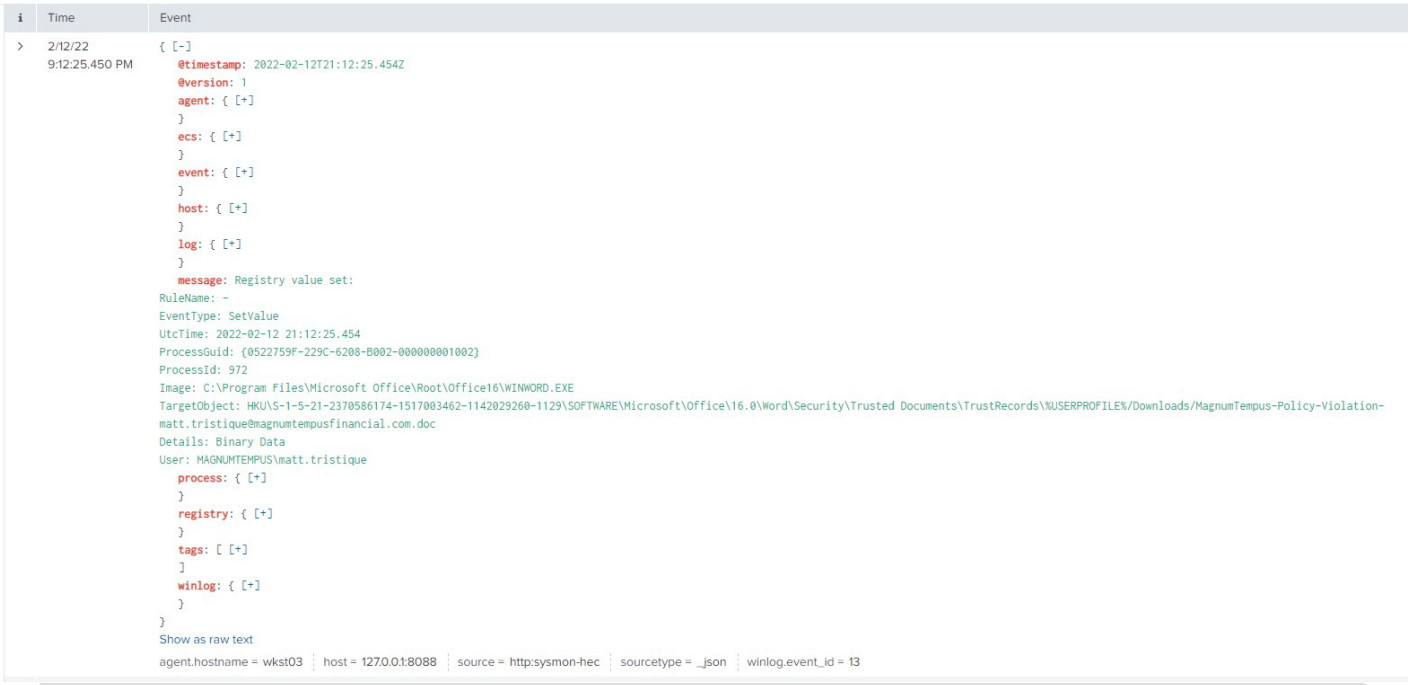


Let's compare with *Excluding* the fileshare from the results:



22 events!

Here's the first event that shows up:



> Registry value set:
> RuleName: -
> EventType: SetValue
> UtcTime: 2022-02-12 21:12:25.454
> ProcessGuid: {0522759F-229C-6208-B002-000000001002}
> ProcessId: 972
> Image: C:\Program Files\Microsoft Office\Root\Office16\WINWORD.EXE
> TargetObject: HKU\S-1-5-21-2370586174-1517003462-1142029260-1129\SOFTWARE\Microsoft\Office\16.0\Word\Security\TrustedDocuments\TrustRecords%USERPROFILE%/Downloads/MagnumTempus-Policy-Violation-matt.tristique@magnumtempusfinancial.com.doc
> Details: Binary Data
> User: MAGNUMTEMPUS\matt.tristique

Based on the above we can see that a file was indeed executed according to the "Trusted Documents\TrustRecords" we saw in the SANS example of macro activity. This is a step in the right direction.

There were other users with similar activity, let's take a look:

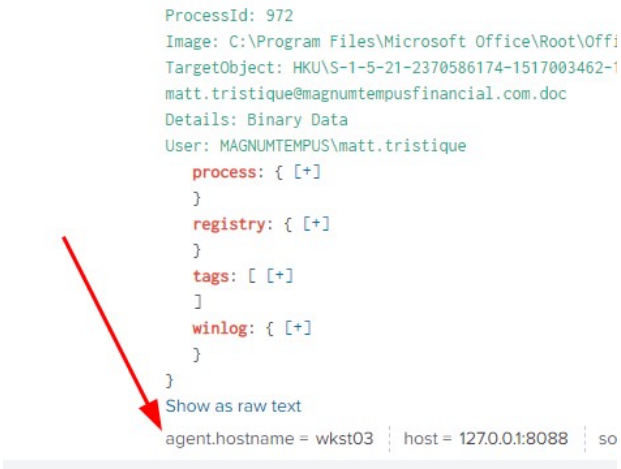
> TargetObject: HKU\S-1-5-21-2370586174-1517003462-1142029260-1128\SOFTWARE\Microsoft\Office\16.0\Word\Security\Trusted Documents\TrustRecords
> %USERPROFILE%/Desktop/MagnumTempus-Policy-Violation-karen.metuens@magnumtempusfinancial.com.doc
> TargetObject: HKU\S-1-5-21-2370586174-1517003462-1142029260-1126\SOFTWARE\Microsoft\Office\16.0\Word\Security\Trusted Documents\TrustRecords
> %USERPROFILE%/Desktop/MagnumTempus-Policy-Violation-amanda.nuensis@magnumtempusfinancial.com.doc

All of the filetypes appear to be .doc and the filenames appear to be extremely formulaic and similar:

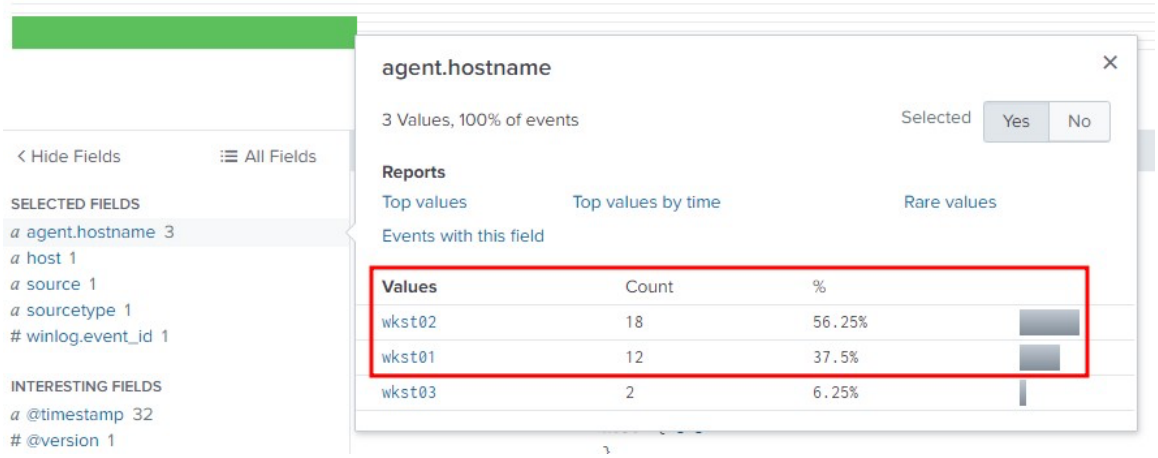
> MagnumTempus-Policy-Violation-karen.metuens3@magnumtempusfinancial.com.doc

This is an interesting filename because it appears to include an email in the filename, which is NOT normal and could be a sign of a malicious file.

If we look at the bottom of the first entry, we see a workstation name (agent.hostname):



Let's see if Splunk can show us more data. On the left side of the page, go to the "SELECTED FIELDS" section, then click on "agent.hostname":



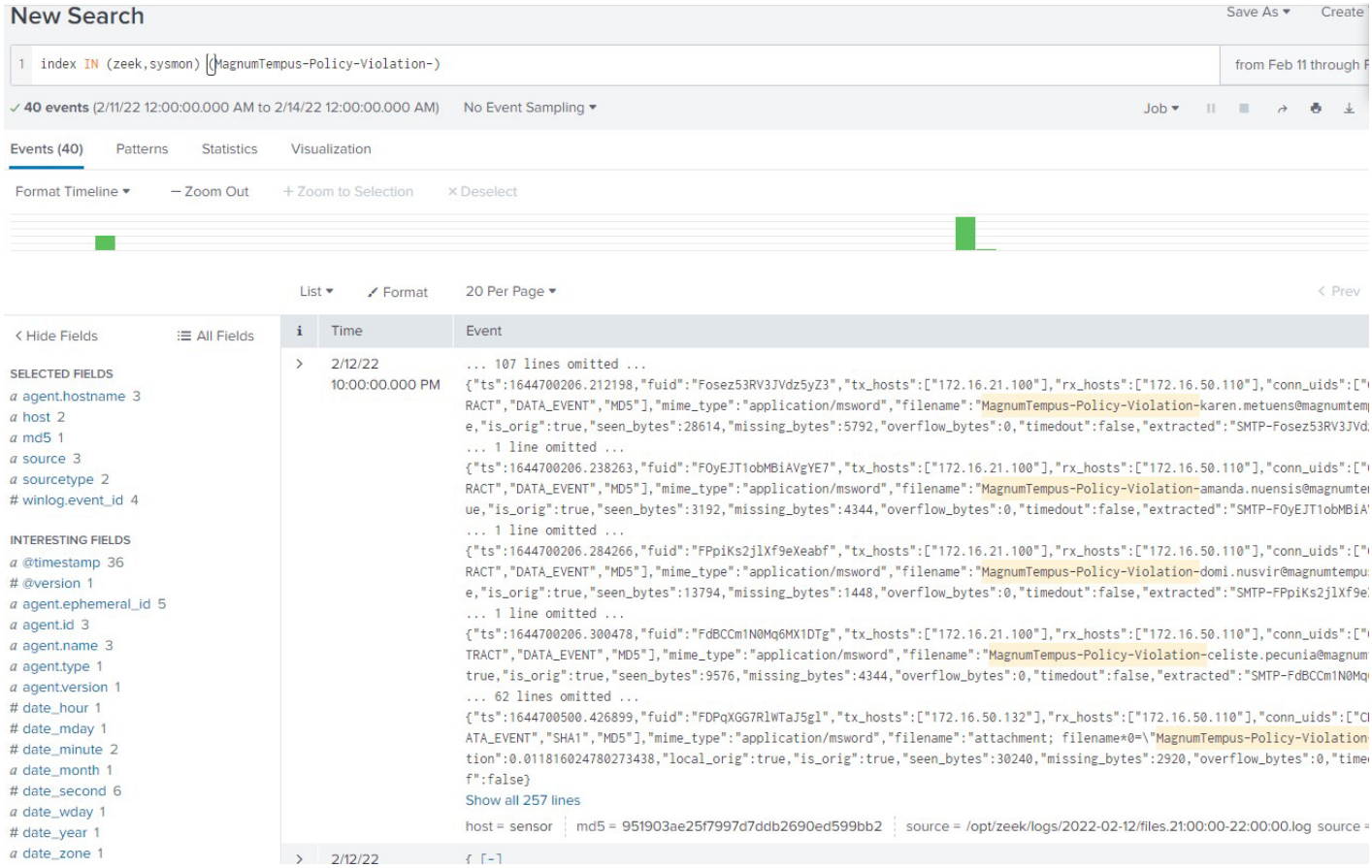
Matt's hostname appears to be wkst03 - not many hits. wkst01 and wkst02 have a significantly higher number of entries, and this could signify additional activity.

Can we find the source of the files?

Let's try to see what we can find with this filename in the logs. We have to create a new query:

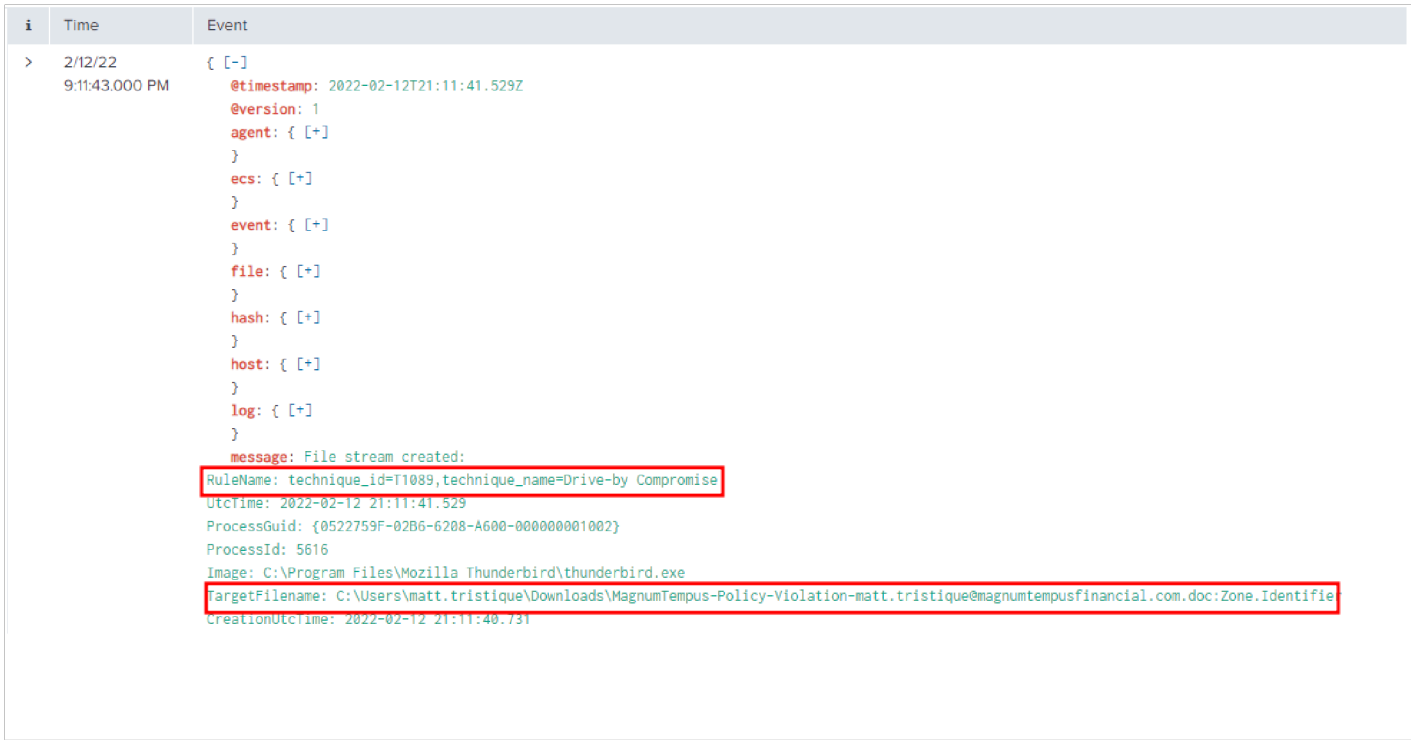
```
index IN (zeek,sysmon) (MagnumTempus-Policy-Violation-)
```

When we run the query, we see additional users were targeted:



MagnumTempus-Policy-Violation-domi.nusvir@magnumtempusfinancial.com.doc
MagnumTempus-Policy-Violation-celiste.pecunia@magnumtempusfinancial.com.doc

We also see where the file was downloaded from for one user, Matt Tristique:



> File stream created:
> RuleName: technique_id=T1089,technique_name=Drive-by Compromise
> UtcTime: 2022-02-12 21:11:41.529
> ProcessGuid: {0522759F-02B6-6208-A600-000000001002}
> ProcessId: 5616
> Image: C:\Program Files\Mozilla Thunderbird\thunderbird.exe
> TargetFilename: C:\Users\matt.tristique\Downloads\MagnumTempus-Policy-Violation-matt.tristique@magnumtempusfinancial.com.doc:Zone.Identifier
> CreationUtcTime: 2022-02-12 21:11:40.731
> Hash:
> SHA1=AE356A67D337AFA5933E3E679E84854DEEACE048,MD5=DCE5191790621B5E424478CA69C47F55,SHA256=86A3E68762720ABE870D139679485022093511 5D3CCC8BB134FFA521244E3EF8,IMPHASH=00000000000000000000000000000000
> Contents: [ZoneTransfer] ZoneId=3 HostUrl=about:internet
> User: MAGNUMTEMPUS\matt.tristique

This entry is very intriguing:

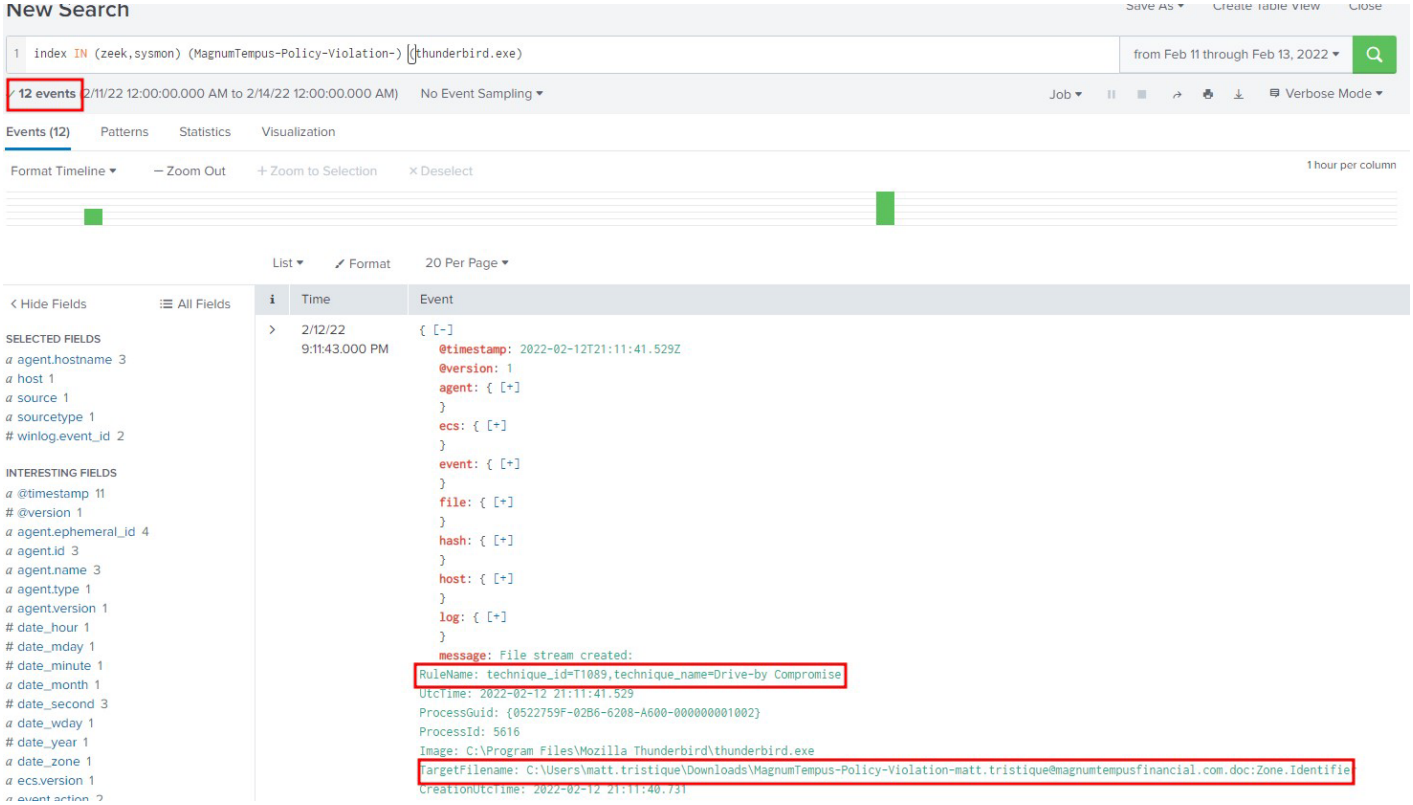
> RuleName: technique_id=T1089,technique_name=Drive-by Compromise

It appears that this file was detected as a *Drive-by Compromise*. This is not good. We might be able to surmise this is a malicious document based on this.

Who else downloaded the document?

Let's see if any other users downloaded this file with ThunderBird:

```
index IN (zeek,sysmon) (MagnumTempus-Policy-Violation-) (thunderbird.exe)
```



From our search, it appears both Amanda and Karen downloaded the file via ThunderBird.

Interestingly enough, neither Amanda's nor Karen's entries signify that the file was detected as a *Drive-by Compromise*:

> File created:
> RuleName: -
> UtcTime: 2022-02-12 21:11:24.552
> ProcessGuid: {29C462BB-0EC0-6208-D000-000000001202}
> ProcessId: 3184
> Image: C:\Program Files\Mozilla Thunderbird\thunderbird.exe
> TargetFilename: C:\Users\amanda.nuensis\Desktop\MagnumTempus-Policy-Violation-amanda.nuensis@magnumtempusfinancial.com.doc
> CreationUtcTime: 2022-02-12 21:11:24.552
> User: MAGNUMTEMPUS\amanda.nuensis

> File created:
> RuleName: -
> UtcTime: 2022-02-11 04:51:45.698
> ProcessGuid: {444CBE19-EAF9-6205-E600-000000001302}
> ProcessId: 5552
> Image: C:\Program Files\Mozilla Thunderbird\thunderbird.exe
> TargetFilename: C:\Users\karen.metuens\Desktop\MagnumTempus-Policy-Violation-karen.metuens@magnumtempusfinancial.com.doc
> CreationUtcTime: 2022-02-11 04:51:45.572
> User: MAGNUMTEMPUS\karen.metuens

What does this mean?

Unfortunately, at least three of Magnum Tempus employees downloaded (and based on what we know about the registry changes involving "Trust Records" - successfully ran the malicious code) a malicious document laden with VBA Macro documents. At this point, it might be worth it to investigate further what has happened as a result of detonation of the intial malicious payload.

Are we done?

In theory, at this point we are done as we have fulfilled the initial hypothesis of determining that a user downloaded and executed the VBA Macro. While we have not necessarily confirmed that the macro code is indeed malicious, we would need to take additional steps to do this.

Can we find the initial email that contained the malicious document - and possibly the document itself and inspect the code?

Yes! However, we will need to use another tool, WireShark in order to extract information from the captured network traffic (PCAP File).

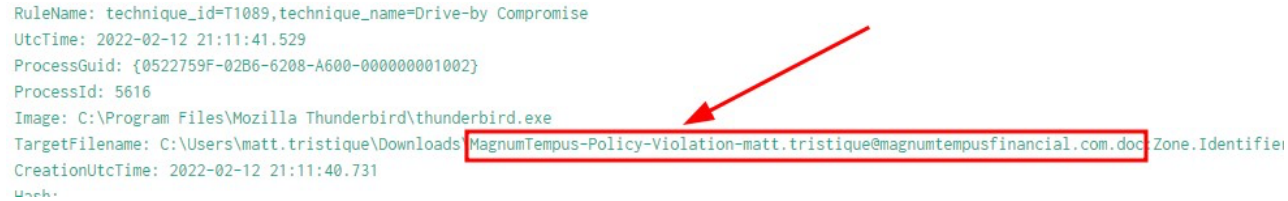
Threat Hunting and Investigation with WireShark

What is WireShark?

WireShark is a specialized tool for analyzing network data. We can review data flowing through the network in real-time, however in this specific case we will be reviewing a point-in-time capture of Magnum Tempus network traffic.

Locating the offending email in WireShark

Since we know what the filename, based on our searches above:



Let's look at the filename:

> MagnumTempus-Policy-Violation-karen.metuens@magnumtempusfinancial.com.doc

Since MagnumTempus is something that we might expect to see in other documents, let's disregard this for the moment. The term "Policy-Violation" stands out as something that might be unique in an organization. Taking this mindset, let's use the following for our further search:

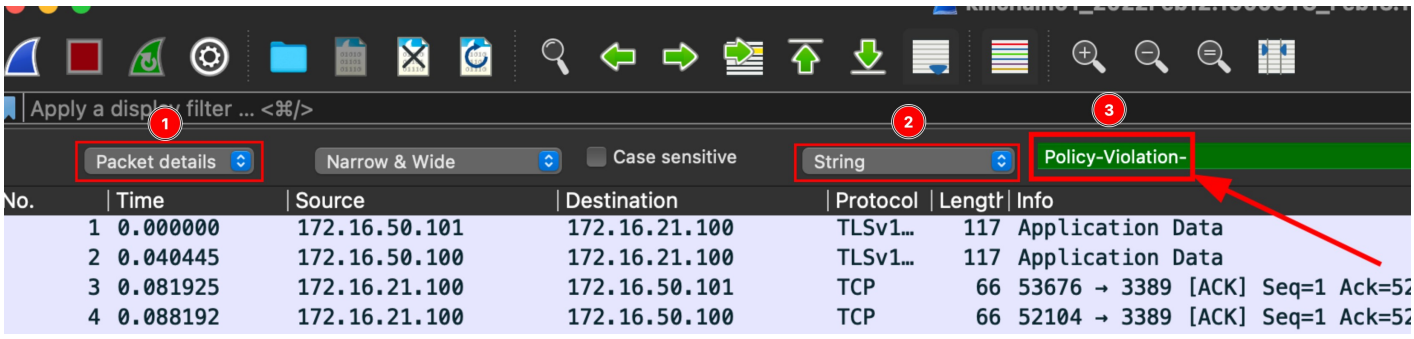
> Policy-Violation-

Now let's check to see if we can find this in WireShark:

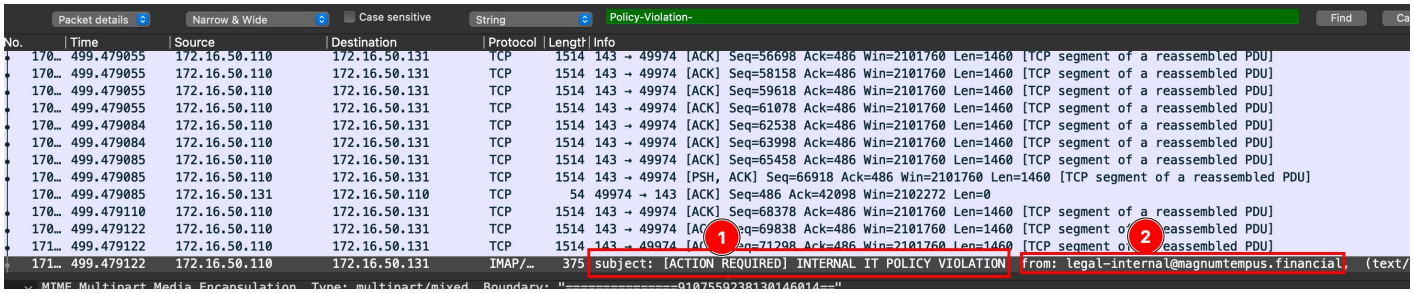
Press control and F to bring up the search tool.

Change the first dropdown to "Packet Details" (1).
In the third dropdown change to "String" (2)
In the text box, type the following (3):

> Policy-Violation-



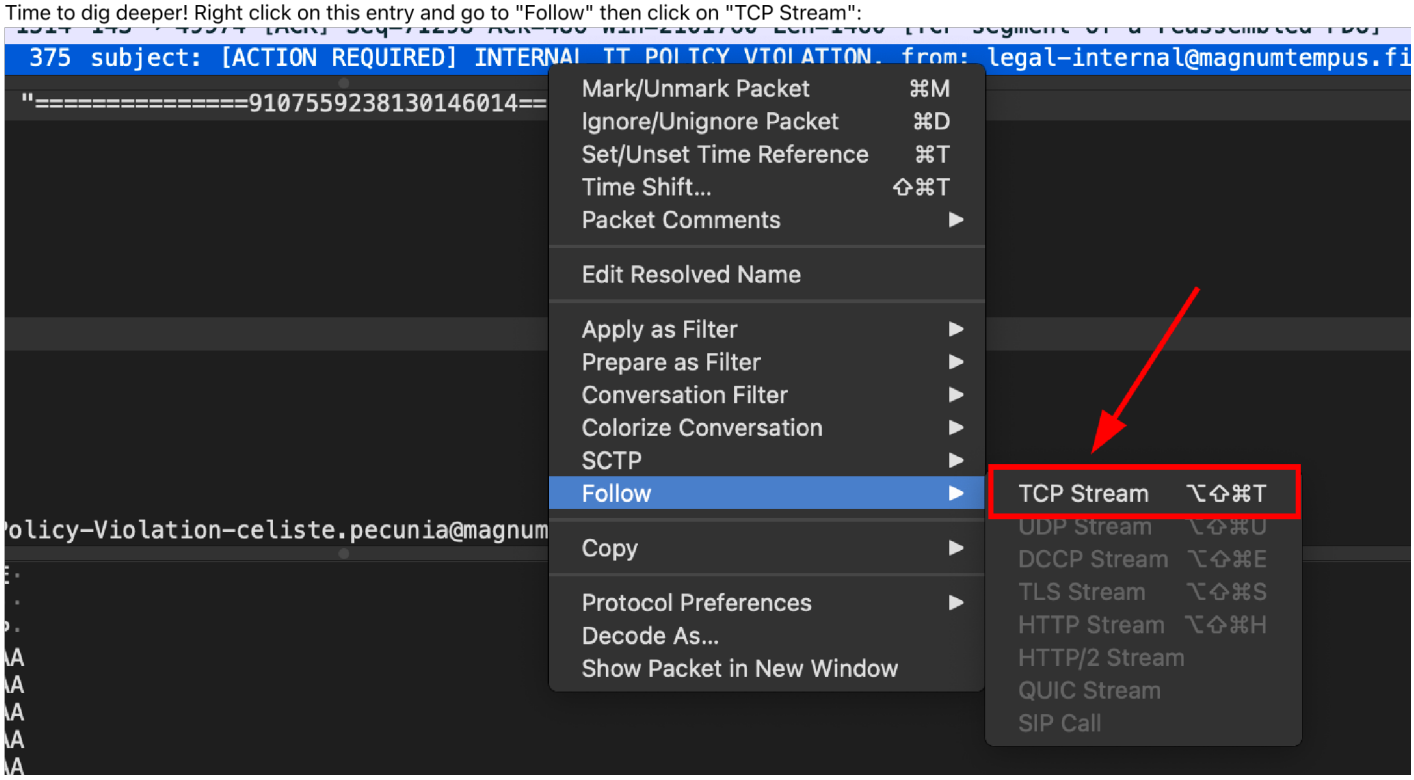
Press "Find":



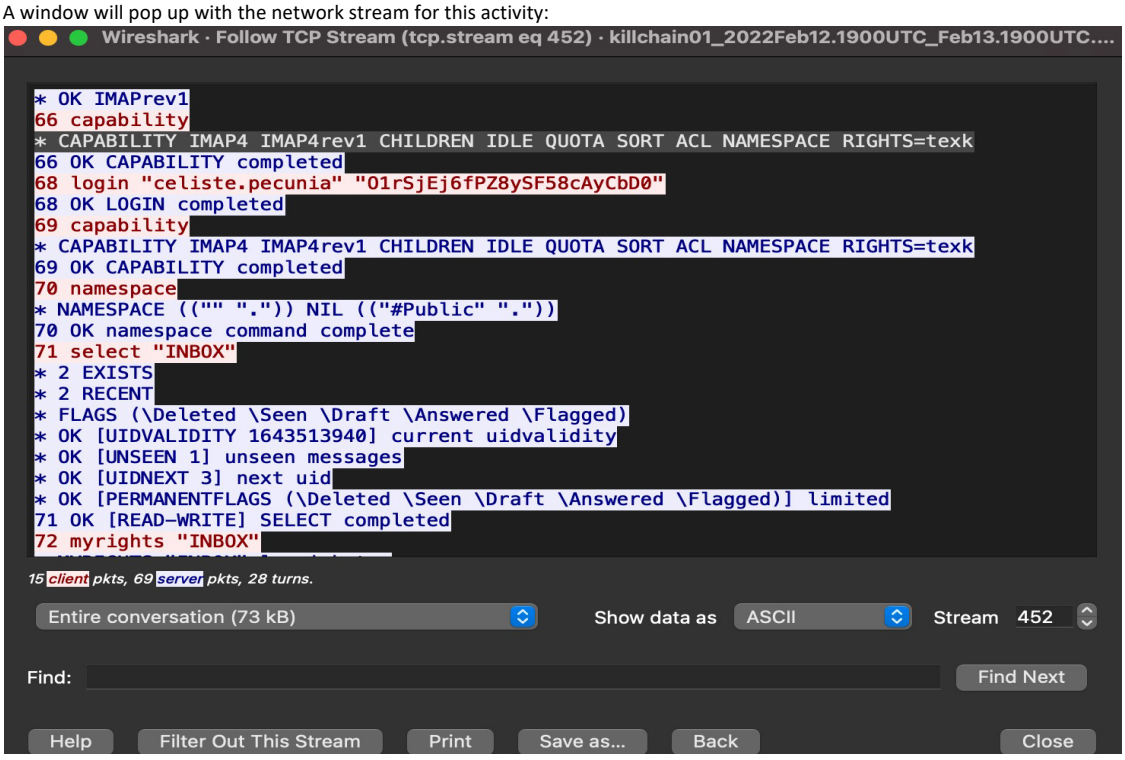
Let's review what we see here:

1. Subject: [ACTION REQUIRED] INTERNAL IT POLICY VIOLATION
2. From: legal-internal@magnumtempus.financial

Time to dig deeper! Right click on this entry and go to "Follow" then click on "TCP Stream":



A window will pop up with the network stream for this activity:



Scrolling down further we get to the content of the email itself. What makes this interesting is that perhaps the mail server was not properly configured, because we are able to see in clear-text over the network contents of an email. This has other security implications that we will not discuss here, but is important to note for future investigations.

We can see the content here:

```
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit


The MagnumTempus Financial CERT and CyberSecurity team have noticed that you are one of the users -
"karen.metuens@magnumtempus.financial", "amanda.nuensis@magnumtempus.financial", who have violated the
company policy CCG-IV:5-8 on 2/7/2022, 8:48pm - EDT.

As mentioned in the yearly cybersecurity training and your employment agreement with MagnumTempus, the
violation of IT policy may terminate your employment.

Please review the attachment which includes the decision made by the MagnumTempus Legal team. Make sure
to reply to this email within 72 hours of opening the document.

Thank you,
MagnumTempus Internal Legal Department
(+1)969-555-5984
legal-internal@magnumtempus.financial
```


```
The MagnumTempus Financial CERT and CyberSecurity team have noticed that you are one of the users -
"karen.metuens@magnumtempus.financial", "amanda.nuensis@magnumtempus.financial", who have violated the company policy CCG-IV:5-8
on 2/7/2022, 8:48pm - EDT.


As mentioned in the yearly cybersecurity training and your employment agreement with MagnumTempus, the violation of IT policy may
terminate your employment.

Please review the attachment which includes the decision made by the MagnumTempus Legal team. Make sure to reply to this email
within 72 hours of opening the document.

Thank you,
MagnumTempus Internal Legal Department
(+1)969-555-5984
legal-internal@magnumtempus.financial
```


Scrolling downfurther, we see there is an attachment:

A few notes about this screenshot:

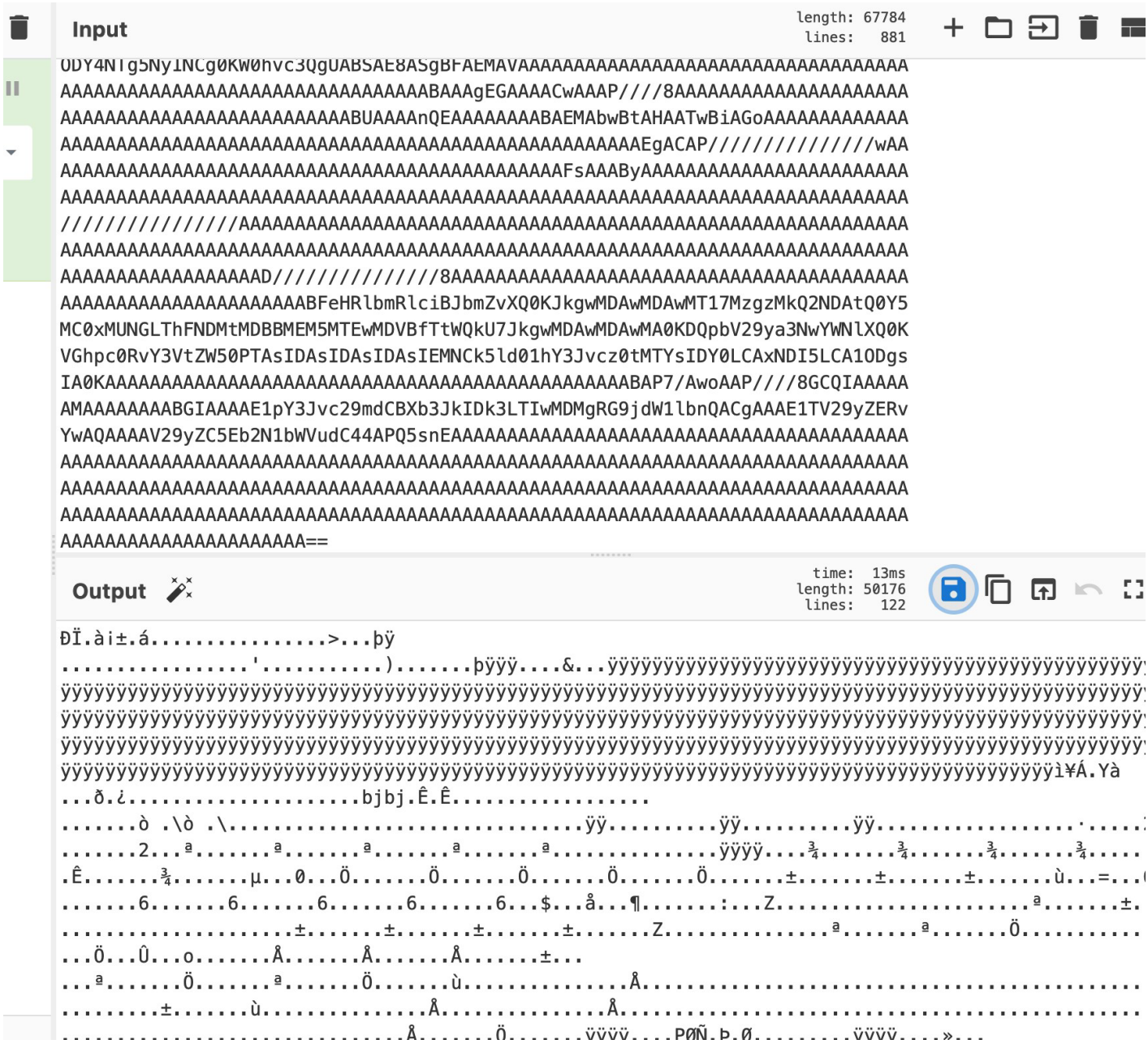
1. The attachment is encoded, this one in Base64. Base64 is a common encoding algorythm that can be used to 'encrypt' documents to transmit.
2. The filename for the attachment matches what we expect to see, based on our findings in our Splunk investigations . This is
3. the Base64 encoded data for the document.

This matches what we saw in our earlier Splunk queries. We might be able to extract the file data from WireShark to analyze further - an important thing to note from above is that the data is encoded in Base64:

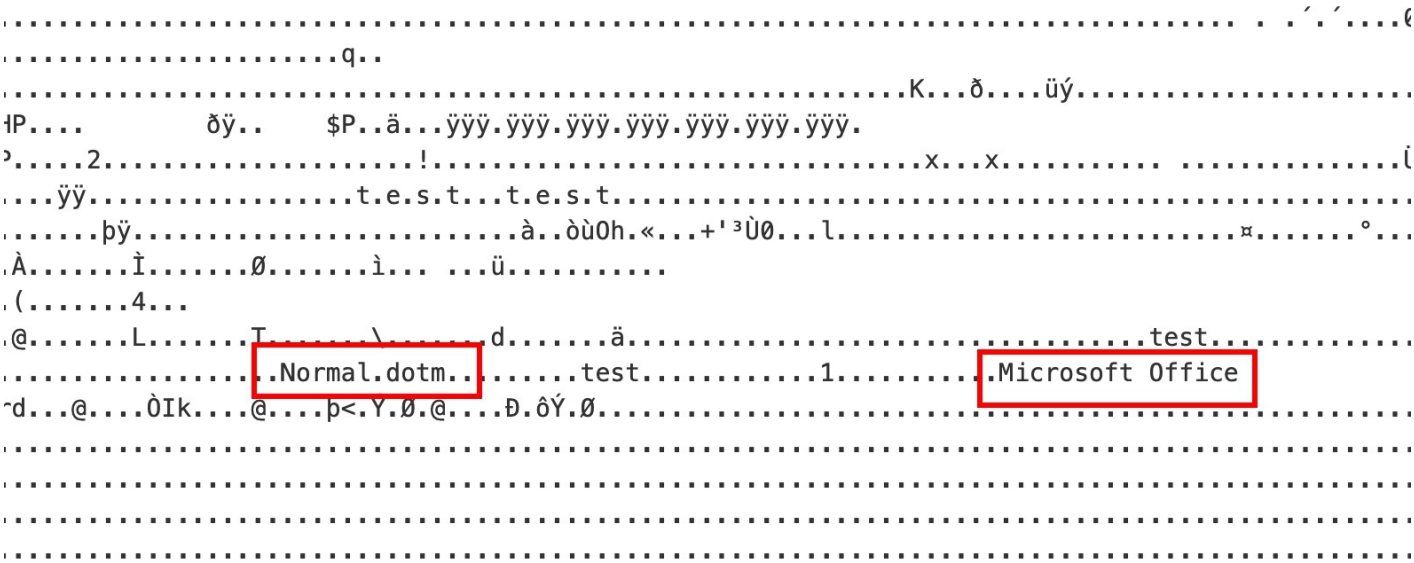
```
NOTE: The below is a truncated version of the actual payload data for brevity purposes:

0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAABAAAAJwAAAAAAAAAA
EAAAKQAAAAEAAAD+////AAAAACYAAAD/////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
pcEAWeAJBAAA8BK/AAAAAAAAEAAAAAAACAAAAQgAAA4AYmpiapDKkMoAAAAAAAAAAAAAAAAAAAAA
AAAJBBYALg4AAPKgDFzyoAxcAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAD//w8AAAAA
AAAAAAD//w8AAAAAAAAAAAD//w8AAAAAAAAAAAAAAAAAAAAAAAAALcAAAAADIHAAAAAAAAMgcAAKoU
AAAAAAAAqhQAAAAAAACqFAAAAAAAAKoUAAAAAAAAqhQAABQAAAAAAAAAAAAAAP////8AAAAAvhQA
AAAAAAC+FAAAAAAAAL4UAAAAAAAAvhQAAAwAAADKFAAADAAAAL4UAAAAAAAAtRcAADABAADWFAAA
AAAAAAAAAAAAAAAAAAAAAAABFeHRlbmRlciBJbmZvXQ0KJkgwMDAwMDAwMT17MzgzMkQ2NDAtQ0Y5
MC0xMUNGLThFNDMtMDBBMEM5MTEwMDVBfTtWQkU7JkgwMDAwMDAwMA0KDQpbV29ya3NwYWNlXQ0K
VGhpc0RvY3VtZW50PTAsIDAsIDAsIEMNCk5ld01hY3Jvcz0tMTYsIDY0LCAxNDI5LCA1ODgs
IA0KAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAP7/AwoAAP////8GCQIAAAAA
AMAAAAAAAABGIAAAAE1pY3Jvc29mdCBXb3JkIDk3LTIwMDMgRG9jdW1lbnQACgAAAE1TV29yZERv
YwAQAAAAV29yZC5Eb2N1bWVudC44APQ5snEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAA==
```

We can then take the data we've copied and then use a tool like CyberChef to convert from Base64 to somewhat human-readable text:



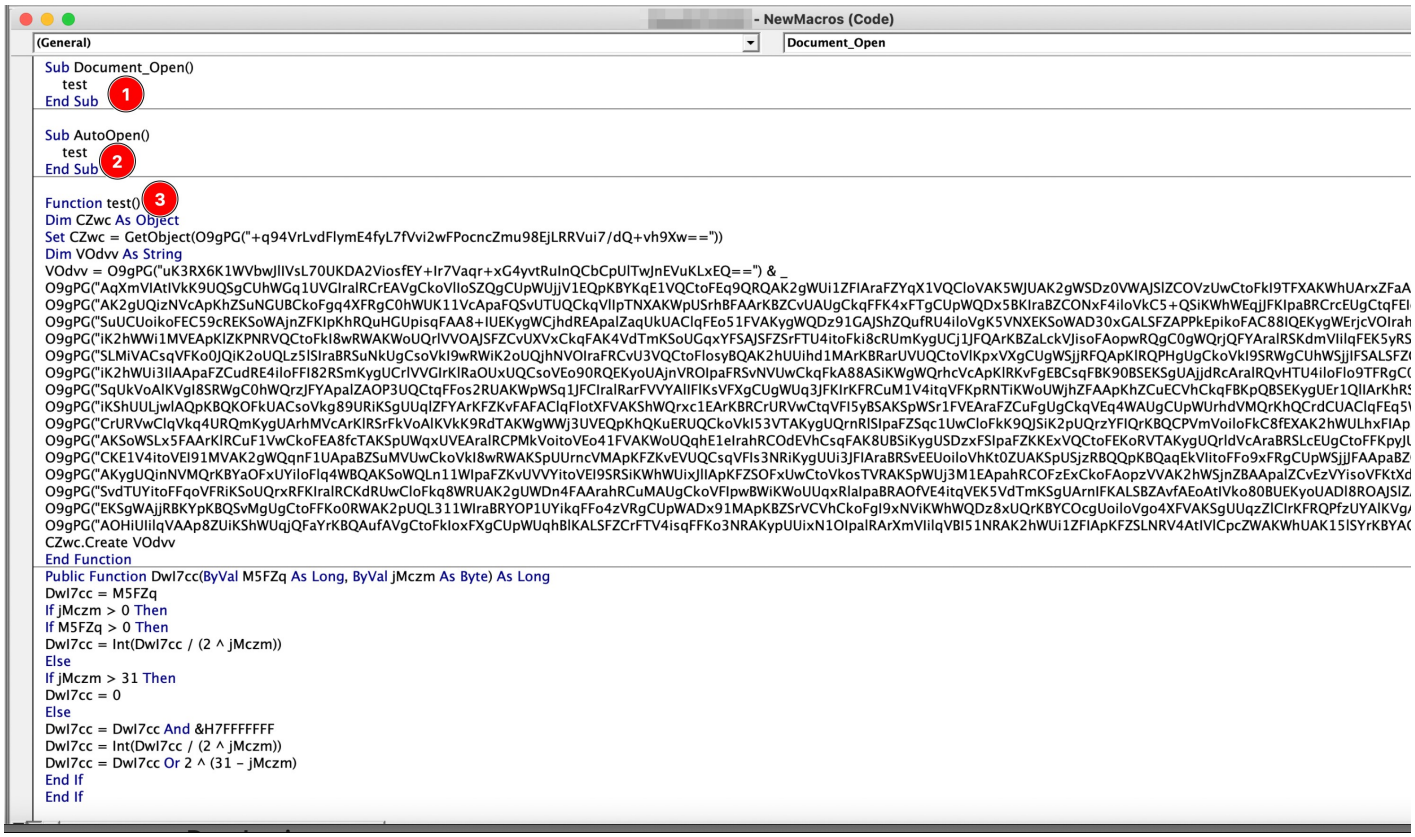
Scrolling down, we can see data inside the converted code that indicates it's a Microsoft Document:



Further down the code, we see proof of Visual Basic for Application Macros:

```
..      ..*.\C..
.Xðîc....!Offic..gO.f..i.c..g¤....!G{2.DF8D04C-.5BFA-101@B-BDE5.gAjA.e4..2.g.ºg.ram File.s
(x86)\@Common.  \.Microsof.t Shared.\OFFICE1.5\MSO.DL.L#...P 15..0 Ob.æ
Li`brary.'.....|...Â..ý..ÂfThisDoc.umentG...À
T@hi.s.D.@Jc.uÀJe.nU@p.Î.2Ú..À..T.HB.1Â.ÿ@...B...,Â!D®"B..+B....`NewDMa.>G....N..Yw.M.lc.rÑ.%s...
        2.      0 .A4..M sh!.Â.C
.Â.....................................................ThisDocument.T.h.i.s.D.o.c.u.m.e.n
.t...NewMacros.N.e.w.M.a.c.r.o.s.....................................................ID="
{7EF64435-84A1-4A2B-B805-A65682CD1FE6}"
Document=ThisDocument/&H00000000
Module=NewMacros
Name="Project"
HelpContextID="0"
VersionCompatible32="393222000"
CMG="52508D34AC38AC38AC38AC38"
DPB="6D6FB255D671D771D771"
GC="888A576858685897"
```

Then, with the assistance of MOVIE MAGIC (I made that up), we are able to reassemble the malicious document to see there are macros with obfuscation:



If we look at the Macro above - there are a few important parts that stick out:

. This is set to automatically run the "test" function/macro on document open.

. This is also set to automatically run the "test" function.

. This macro is heavily encoded, perhaps to hide the true intent behind the function?

This is DEFINITELY NOT GOOD

There should be no reason for this level of encoding for a document of this nature.

At this point, it may be a good spot to pass this to the next team to begin their steps.

THREAT HUNTER TEMPLATE

Playbook Title:
Detecting Enterprise Macro Activity from Emails

Mitre Tactic:
T1566, Phishing

Mitre Sub Technique:

T1566.001, Spearphishing Attachment

Hypothesis:
Employees are targeted with malicious documents with VBA Macro Code and some employees will open the documents and detonate the payload

Proposed Detection Query:

> Leverage Zeek and Sysmon telemetry logs to detect TrustRecords in Microsoft Document Formats

```
index IN (zeek,sysmon) (.doc OR .xls OR .docx OR .xlsx) (TrustRecords)
AND NOT (files.magnumtempusfinancial.com)
```

Simulation Details: NONE

Hunter Limitations/Observation Notes:
During several portions of the hunt, we discovered that there were log sources (sysmon) that were not properly parsed, which made finding details difficult. If we had these parsed properly, we may have found it easier to get some of the data.

Hunt Findings:
Three users downloaded the malicious document, two users appear to have been affected by the payload.