EMERALDWHALE: 15k Cloud Credentials Stolen in Operation Targeting Exposed Git Config Files

The Sysdig Threat Research Team (TRT) recently discovered a global operation, EMERALDWHALE, targeting exposed Git configurations resulting in more than 15,000 cloud service credentials stolen. This campaign used multiple private tools that abused multiple misconfigured web services, allowing attackers to steal credentials, clone private repositories, and extract cloud credentials from their source code. **Credentials for over 10,000 private repositories were collected during the operation.** The stolen data was stored in a S3 bucket of a previous victim.

The stolen credentials belong to Cloud Service Providers (CSP), Email providers, and other services. Phishing and SPAM seem to be the primary goal of stealing the credentials. The credentials themselves can be worth hundreds of dollars per account. The accounts themselves are not the only way EMERALDWHALE make money; the target lists they develop can also be sold on various marketplaces.

This attack shows that secret management alone is not enough to secure an environment. There are just too many places credentials could leak from.

Initial discovery from S3

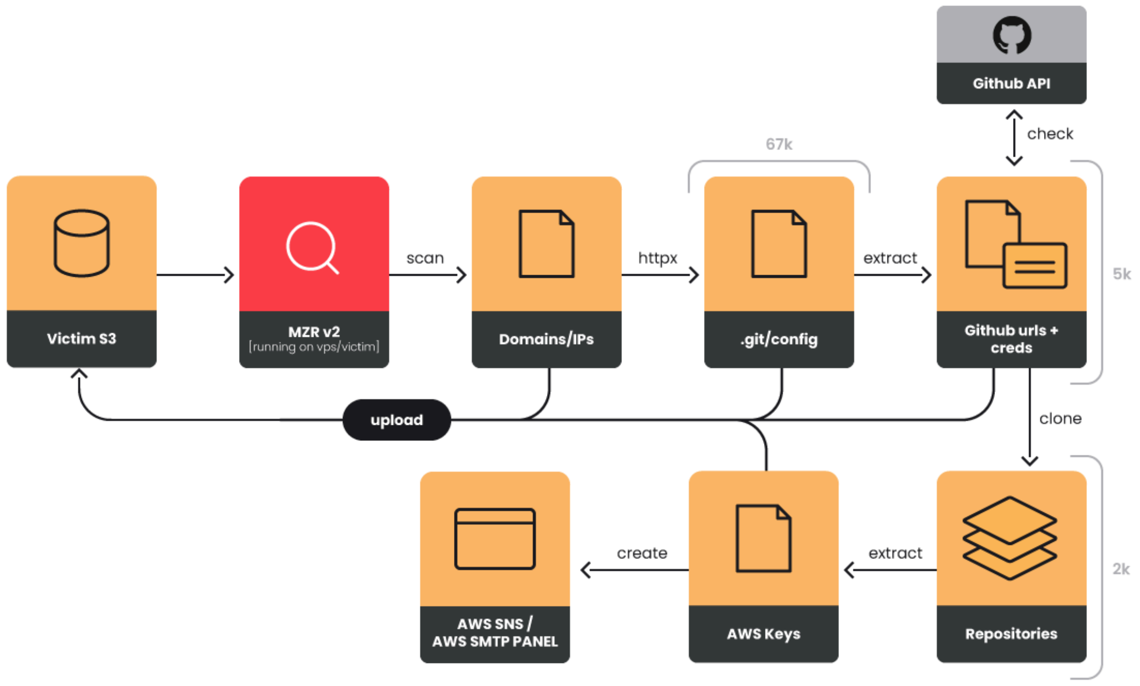
While monitoring the Sysdig TRT cloud honeypot, we observed an unusual *ListBuckets* call using an account that had been compromised. The S3 bucket, *s3simplisitter*, that was referenced did not belong to our account. Instead, it belonged to an unknown account and was publicly exposed. While investigating this bucket, we discovered malicious tools and over a terabyte of data, which included compromised credentials and logging data. **Analysis of the malicious tools revealed a multi-faceted attack, including web scraping Git config files, Laravel *.env* files, and raw web data.**

We reached out to AWS to report the bucket, which they took down.

Git Configuration Exploitation

The log data recovered from the bucket showed a massive scanning campaign between August and September for servers that had exposed Git repository configuration files. EMERALDWHALE targeted large swaths of the Internet as scanning on this scale has become easier with open-source tools, such as httpx.

Below is an outline of the attack chain:



Starting with long lists of IP address ranges, the toolset used by EMERALDWHALE automatically discovers relevant hosts, extracts credentials, and validates the recovered tokens. It then uses the stolen tokens to clone repositories, both public and private, belonging to any Git-compatible service. The tool scans the downloaded repositories and extracts more credentials. Finally, all of the results are uploaded to the S3 bucket.

Why Git Configurations?

Git is a Concurrent Versions System (CVS) that allows developers to work on the same code base and allows for the management and deployment of software projects. GitHub is currently the most popular example of a service that uses the Git protocol. Many tools enable the use of these services — Git is a popular one for Linux and command-line users. The tool will store any setting and authentication in a configuration file.

The *.git* directory contains all information required for version control, including the complete commit history, configuration files, branches, and references. If the *.git* directory is exposed, attackers can retrieve valuable data about the repository's history, structure, and sensitive project information. This includes commit messages, usernames, email addresses, and passwords or API keys if the repository requires them or if they were committed.

One frequent method of exposing the *.git* directory is through web server misconfigurations. If web server permissions are not properly set, users may be able to directly access the *.git* directory via the web, enabling them to download the entire repository and analyze the exposed content. EMERALDWHALE abused this security problem to scan the repository for exposed credentials, collect them, and then sell or use them for other purposes.

Collecting and abusing credentials from public Github repositories has become less effective. Companies will regularly scan these repositories and flag any discovered credentials. AWS, for example, will proactively attach a policy to the credentials that quarantine the keys, limiting their abuse potential.

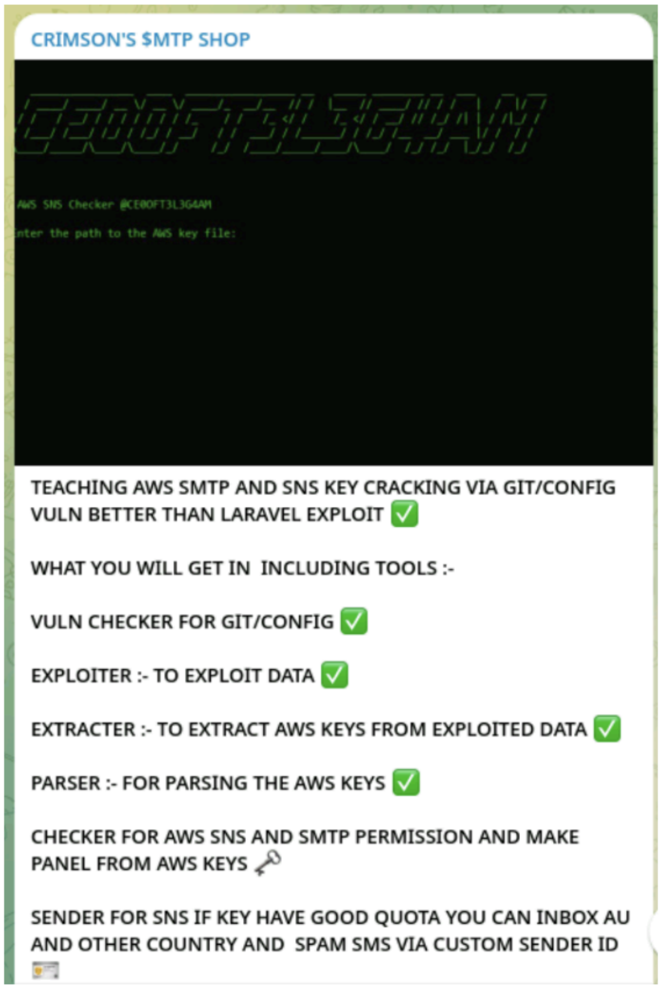
EMERALDWHALE Tools

During our investigation, we found two tools related to vulnerability scanning and exploitation of exposed Git repositories:

- MZR V2 (MIZARU) by @kosov2
- Seyzo-v2

These tools are often sold in underground marketplaces. In addition, we are starting to see that not only tools are being offered, but also entire courses on how to use them to create spam or phishing campaigns.

Below are a couple of examples:


CRIMSON'S $MTP SHOP

```
CEOOFT3L3G4AM

AWS SNS Checker @CEOOFT3L3G4AM

nter the path to the AWS key file:
```

TEACHING AWS SMTP AND SNS KEY CRACKING VIA GIT/CONFIG VULN BETTER THAN LARAVEL EXPLOIT ✅

WHAT YOU WILL GET IN INCLUDING TOOLS :-

VULN CHECKER FOR GIT/CONFIG ✅

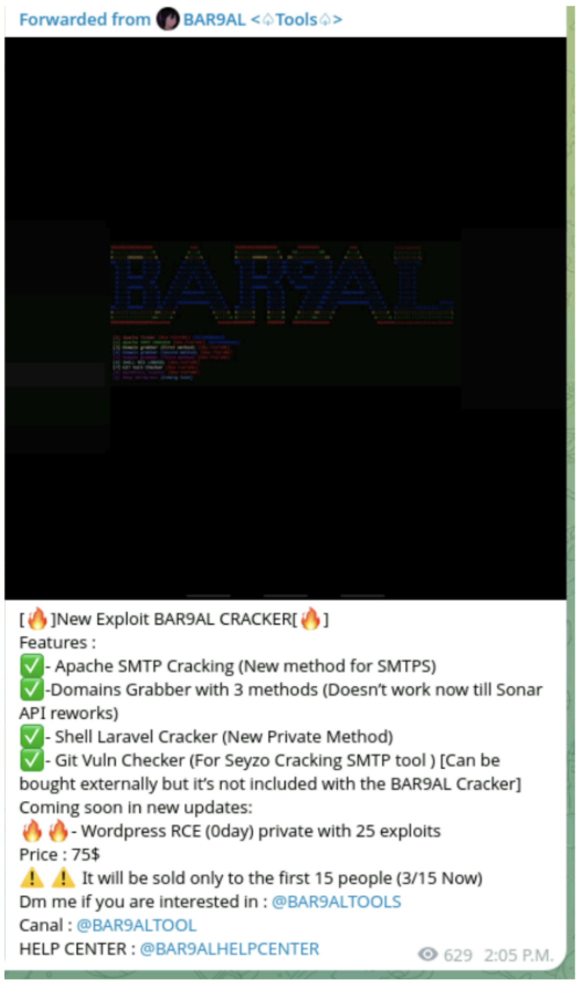
EXPLOITER :- TO EXPLOIT DATA ✅

EXTRACTER :- TO EXTRACT AWS KEYS FROM EXPLOITED DATA ✅

PARSER :- FOR PARSING THE AWS KEYS ✅

CHECKER FOR AWS SNS AND SMTP PERMISSION AND MAKE PANEL FROM AWS KEYS 🔑

SENDER FOR SNS IF KEY HAVE GOOD QUOTA YOU CAN INBOX AU AND OTHER COUNTRY AND SPAM SMS VIA CUSTOM SENDER ID


Forwarded from BAR9AL <Tools>

```
BAR9AL
```

[🔥]New Exploit BAR9AL CRACKER[🔥]
Features :
✅- Apache SMTP Cracking (New method for SMTPS)
✅-Domains Grabber with 3 methods (Doesn't work now till Sonar API reworks)
✅- Shell Laravel Cracker (New Private Method)
✅- Git Vuln Checker (For Seyzo Cracking SMTP tool) [Can be bought externally but it's not included with the BAR9AL Cracker]
Coming soon in new updates:
🔥🔥- Wordpress RCE (0day) private with 25 exploits
Price : 75$
⚠️ ⚠️ It will be sold only to the first 15 people (3/15 Now)
Dm me if you are interested in : @BAR9ALTOOLS
Canal : @BAR9ALTOOL
HELP CENTER : @BAR9ALHELPCENTER
629 2:05 P.M.

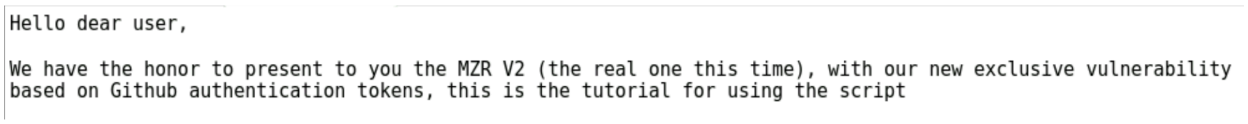
Both MZR V2 and Seyzo-v2 require a list of targets. These lists are usually IPs or domains that have been previously scanned and known to be active. There are several ways to create these target lists. Some common methods are:

- Legitimate Search engines: Google Dorks, Shodan, and other Internet mapping services.
- Scanning tools: Masscan is one of the most used. RUBYCARP for example, uses its botnets to execute scanning and map active IPs.
- Buy it directly from the underground market or data provider.

Let's dig deeper into these tools to understand how they work.

MZR V2 - MIZARU

The discovered tool had a README included with instructions to follow the whole process. This is the only file written in English; the comments in scripts and other files are in French. MZR V2 is made up of a collection of Python scripts and shell scripts.



```
Hello dear user,

We have the honor to present to you the MZR V2 (the real one this time), with our new exclusive vulnerability
based on Github authentication tokens, this is the tutorial for using the script
```

First lines of the Readme

The first script, *gitfinder.sh,* uses the *httpx* tool to scan the target list of IPs. Httpx, which was also used by CRYSTALRAY, is an OSS tool that can scan web servers in a highly parallelized way, making it very efficient.

```
httpx -l $1 -silent -threads 300 -path '/.git/config' -ms '[core]' >> git.txt
```

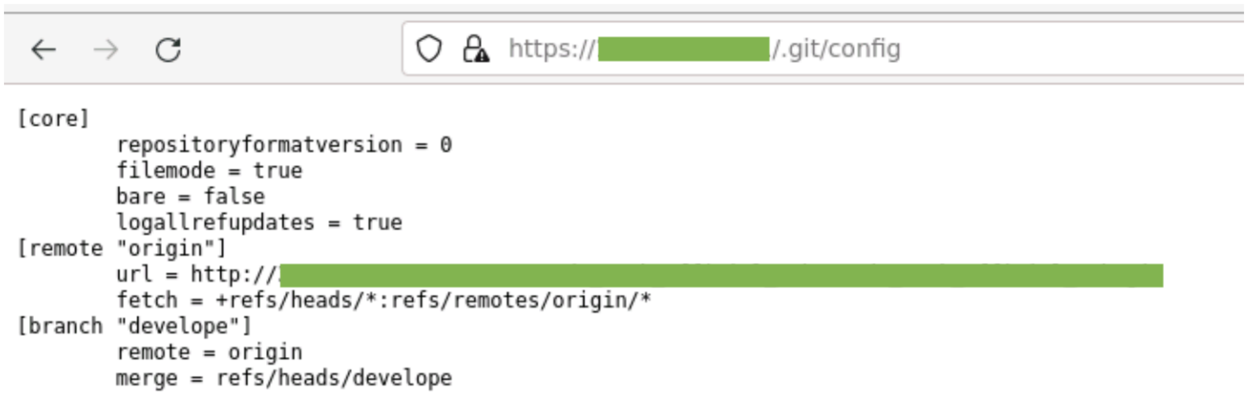
The $1 value contains the input IP addresses. The result is a list containing lines like *https[:]//<IP>/.git/config*.

The second step is to run a Python script, *ghpurl.py,* that makes the query using *wget* and extracts the URL content, using simple regex:

```
match = re.search(r'url = (.+)', content)
```

The extracted URLs are saved to another file for further analysis. An example would be:

```
https://<user>:<token>@<github|gitlab|bitbucket>/<user>/<repo>.git
```



```
https://[redacted]/.git/config

[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[remote "origin"]
        url = http://[redacted]
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "develope"]
        remote = origin
        merge = refs/heads/develope
```

To validate GitHub credentials, the *checkuser.sh* script reaches out to GitHub's API using the information obtained in the previous step. If it is successful, it saves the credential again in a new file. The request to GitHub looks like the following:

```
curl -s https://$fdp@api.github.com/user | jq '.login'
```

With these credentials, the script, *dumpsph.sh*, downloads the repository and extracts the credentials stored in the files using a simple grep.

```
grep -C25 -rPn --exclude='*.html' 'AKIA[A-Z0-9]{16}' .
```

Currently, the tool does not check for old commits or branches, it only checks the current files in the folder when the repository is cloned.

Finally, they have another script (*parser.sh*) that formats the collected data into something more usable by subsequent commands. Here's an example grep to collect AWS keys:

```
grep -aoP '(?<![A-Za-z0-9/+=])[A-Za-z0-9/+=]{40,}(?![A-Za-z0-9/+=])' | head -n1 | sed
-e 's/^\(KEY\|SECRET\)=//g'); region=$(strings $1 | grep -A5 "$i" | grep -aoP
'(us(-gov)?|ap|ca|cn|eu|sa)-(central|(north|south)?(east|west)?)-[0-9]'
```

The last step MZR V2 takes is to use the AWS CLI commands to verify the credentials and check their capabilities. Depending on the option given, new users can be created or additional reconnaissance can be conducted.

1. Check login status and automatically create login credentials.
    a. Use another script, make_panel.sh with "mailer-sns-smtp" as the username, to create the new user and attach the AdministratorAccess policy.
2. Check SMTP permissions and quota and automatically create SMTP credentials.
    a. Convert a Secret Access Key for an IAM user to an SMTP password with ses_password.py.

Once MZR V2 has checked the credentials for SMTP and IAM, it checks for SNS service to see if it can send SMS messages with the *sns_checker.sh* script.

Finally, it uses another series of scripts, one of them in Javascript, which requires the installation of Node and npm, to verify that email sending works. The result is that MZR V2 creates the following two files with the new account information:

- healthy_aws_smtp.txt,

- ses_valid.txt

Seyzo-v2

Similar to MZR V2, Seyzo-v2 is a collection of scripts used to find and steal credentials. There were also several French strings found in the scripts. Seyzo-v2 is started with the *gitfinder.sh* script, which also uses *httpx* to discover exposed Git configuration files and create the target list.

Next, the script *dumperz.sh* used the OSS tool git-dumper to gather all the info from the targeted repositories. This tool is more comprehensive than the methods used in MZR V2.

This is a snippet from the *dumperz.sh* script showing its usage of git-dumper and how it searches the resulting data:

```
git-dumper -j 50 $i $name #git-dumper -h pour savoir comment config proxy, nombre de threads etc

grep --exclude='*.html' -C25 -rPn 'AKIA[A-Z0-9]{16}' --binary-files=text $name/ | cut -c -500

grep -rniP -C25 "smtp\.sendgrid\.net|smtp\.mailgun\.org|smtp-relay\.sendinblue\.com|email-smtp\.(us|eu|ap|ca|cn|sa)-(central|(north|south)?(west|east)?)-[0-9]{1}\.amazonaws.com|smtp.tipimail.com|smtp.sparkpostmail.com|smtp.deliverabilitymanager.net|smtp.mailendo.com|mail.smtpeter.com|mail.smtp2go.com|smtp.socketlabs.com|secure.emailsrvr.com|mail.infomaniak.com|smtp.pepipost.com|smtp.elasticemail.com|smtp25.elasticemail.com|pro.turbo-smtp.com|smtp-pulse.com|in-v3.mailjet.com" --binary-files=text $name | cut -c -500 >> smtp.txt

grep -rniP -C25 "(?i)twilio(.{0,20})?SK[0-9a-f]{32}|nexmo_key|nexmo_secret|nexmo_api" --binary-files=text $name | cut -c -500 >> api_sms.txt
```

As seen above, there are more searches to gather SMTP, SMS, and cloud mail provider credentials. Seyzo-v2 is not entirely focused on stealing CSP credentials like the previous tool. Once it gains access to credentials, it uses the keys in the same way as previously described to create users for SPAM and phishing campaigns.

IoCs

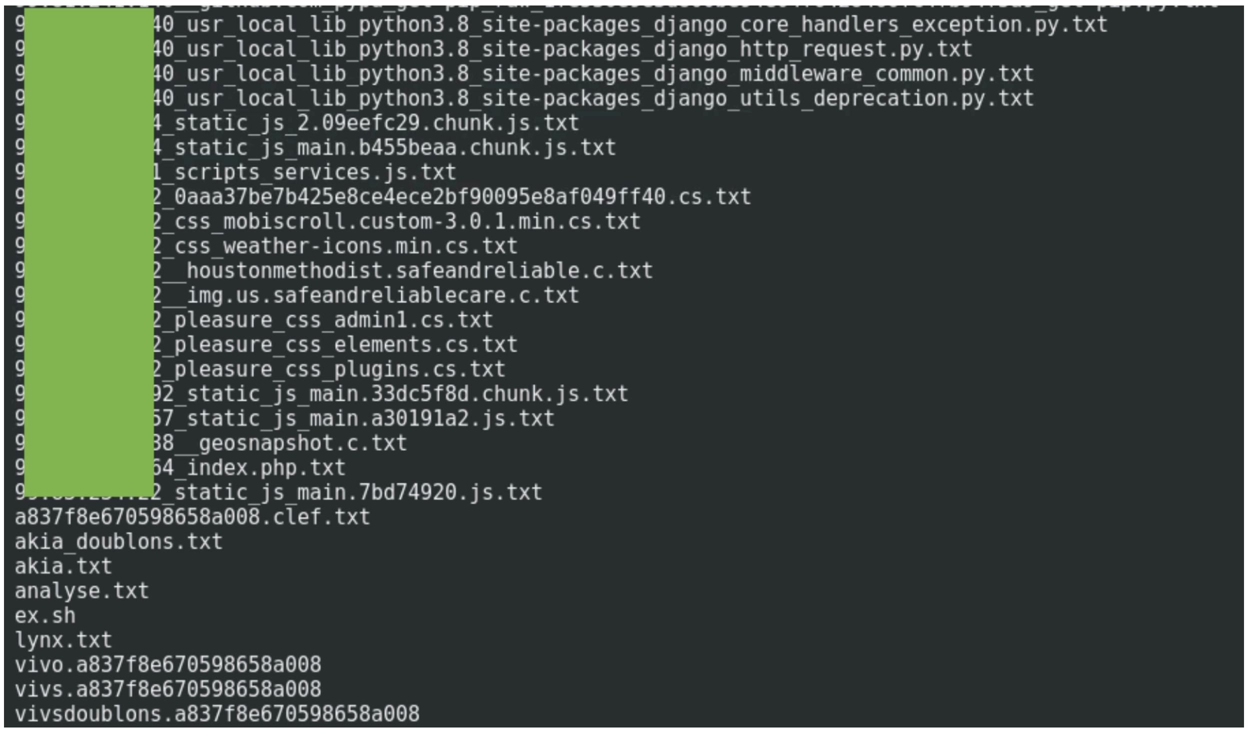
In the table below, we have added the IoCs with AWS CLI commands and keywords used by the tools.

IOCs	
Username	mailer-sns-smtp
Username	mizaruveryhq
Username	s3-admin
Username	SupportAWS
Password	SupportAWS123
Password	@Myregular2910Evolutions@
AWS CLI Command	aws ses get-send-quota
AWS CLI Command	aws ses list-identities
AWS CLI Command	aws sesv2 get-account
AWS CLI Command	aws sts get-caller-identity
AWS CLI Command	aws iam list-users
AWS CLI Command	aws sns get-sms-attributes
AWS CLI Command	aws iam create-user --user-name $username
AWS CLI Command	aws iam attach-user-policy --user-name $username --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
AWS CLI Command	aws iam create-login-profile --user-name $username --password "$password"
AWS CLI Command	aws s3 ls

Raw Web Scraping

We discovered that EMERALDWHALE was not only looking for misconfigured servers and exposed credentials but also had another technique at its disposal. It also used bulk web scraping, followed by extracting cloud credentials in the collected assets. We found dozens of folders with similar names, each containing downloaded assets from the targeted websites. For example, statically defined cloud credentials were found in Javascript files utilized by the website.

In the following image, we have an example of the last files in a folder:



```
9[redacted]40_usr_local_lib_python3.8_site-packages_django_core_handlers_exception.py.txt
9          40_usr_local_lib_python3.8_site-packages_django_http_request.py.txt
9          40_usr_local_lib_python3.8_site-packages_django_middleware_common.py.txt
9          40_usr_local_lib_python3.8_site-packages_django_utils_deprecation.py.txt
9          4_static_js_2.09eefc29.chunk.js.txt
9          4_static_js_main.b455beaa.chunk.js.txt
9          1_scripts_services.js.txt
9          2_0aaa37be7b425e8ce4ece2bf90095e8af049ff40.cs.txt
9          2_css_mobiscroll.custom-3.0.1.min.cs.txt
9          2_css_weather-icons.min.cs.txt
9          2__houstonmethodist.safeandreliable.c.txt
9          2__img.us.safeandreliablecare.c.txt
9          2_pleasure_css_admin1.cs.txt
9          2_pleasure_css_elements.cs.txt
9          2_pleasure_css_plugins.cs.txt
9          92_static_js_main.33dc5f8d.chunk.js.txt
9          57_static_js_main.a30191a2.js.txt
9          38__geosnapshot.c.txt
9          64_index.php.txt
9          2_static_js_main.7bd74920.js.txt
a837f8e670598658a008.clef.txt
akia_doublons.txt
akia.txt
analyse.txt
ex.sh
lynx.txt
vivo.a837f8e670598658a008
vivs.a837f8e670598658a008
vivsdoublons.a837f8e670598658a008
```

We found several standard scripts and output files in each folder that are involved in collecting and analyzing the targeted website's data.

The main file is *ex.sh*. This shell script analyzes collected files looking for and extracting cloud credentials. The regex used are similar to those seen in the other tools.

```
grep -C15 -rPn 'AKIA[A-Z0-9]{16}'
```

```
grep -E -a -o
"(us|eu|ap|ca|cn|sa|me)-(central|(north|south)?(west|east)?(gov-west|gov-east)?)-[0-9]
{1}"
```

```
grep -aoP '(?<![A-Za-z0-9/+=])[A-Za-z0-9/+=]{40,}(?![A-Za-z0-9/+=])'
```

The rest of the files are temporary files generated and deleted once the extraction process is finished. In this specific example, they are shown since they're in a folder with the scraping still active.

Target and Victim Analysis

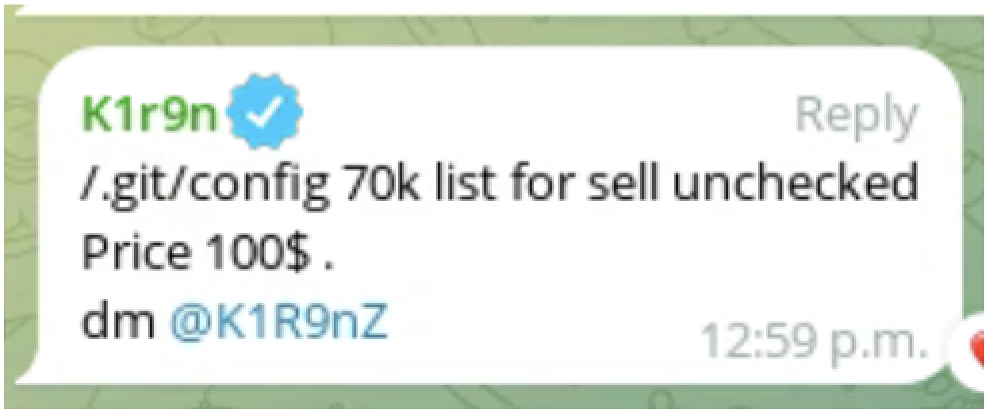
The logging data left in the S3 bucket by EMERALDWHALE allows us to get an idea about the operation's scope and success. The data includes targeting lists, tool output, and raw data collected.

As previously mentioned, the workflow of both tools used lists of targets to start the attack chain. Analysis of the target lists revealed the following:

- IP Addresses: 500M+
- IP Ranges: 12k
- Domains: 500k
- EC2 hostnames: ~1M

Fun fact: they stored a file with all IPV4s one IP per line (1.1.1.1 to 255.255.255.255.255) resulting in 4,278,190,082 lines.

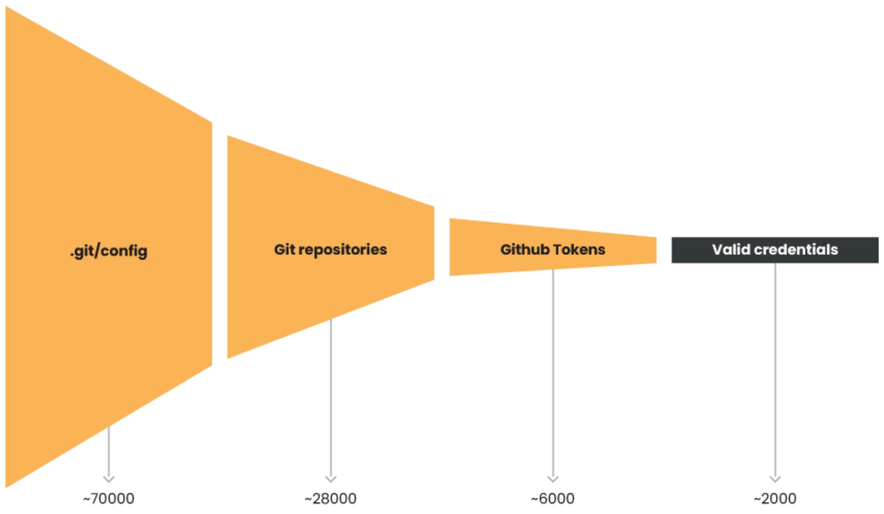
Using one of these target lists, the attackers used the MZR V2 tool and were able to discover more than 67,000 URLs with the path *\/.git/config* exposed. We did some investigation on Telegram and found that the list alone sells for $100. This confirms there is an active market for Git configuration files.



K1r9n
Reply
/.git/config 70k list for sell unchecked
Price 100$.
dm @K1R9nZ
12:59 p.m.

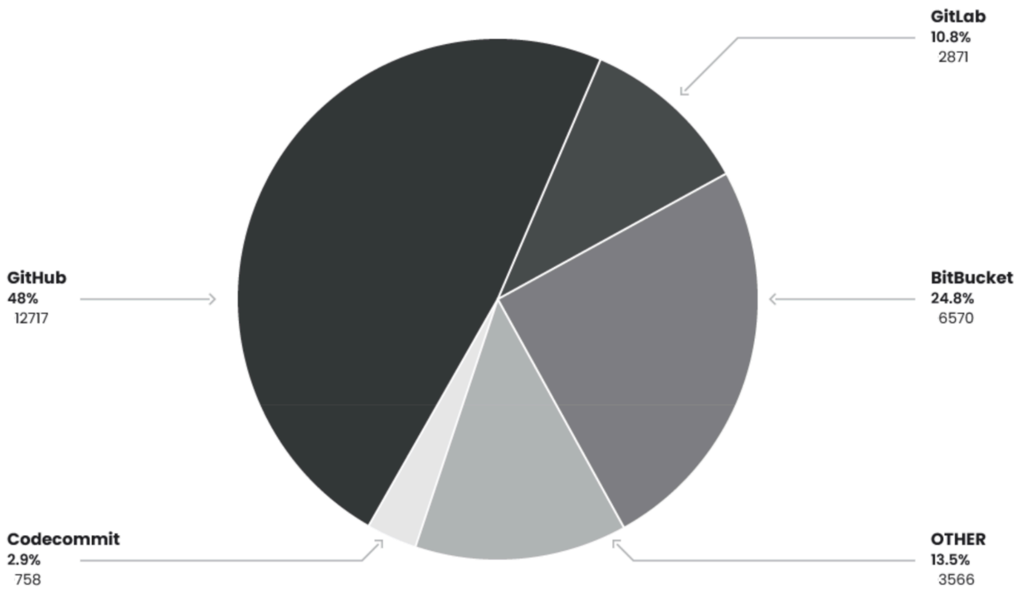
This price may be so high because search engines like Internet mapping services, such as Shodan or Censys, cannot search by URL path. It is possible to find some of these exposed files using Google Dorks but it is difficult to get such a large number of these compared to active scanning.

There were many different repositories collected from all of the exposed Git configuration files. Most belonged to major services such as GitHub, BitBucket, and GitLab. To get a better estimate of how many of the discovered credentials were valid, we conducted limited analysis on the approximately 6,000 GitHub tokens and determined roughly 2,000 were valid credentials.



Funnel of Git repositories to valid credentials

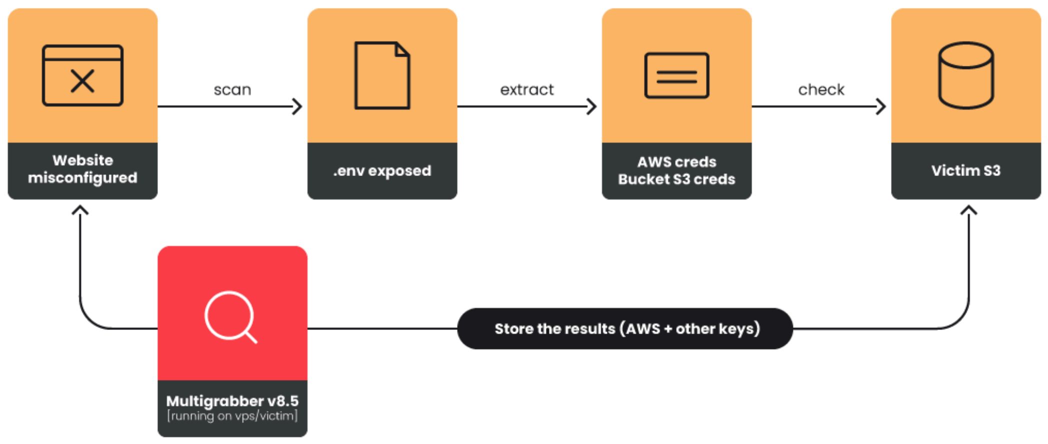
While GitHub, BitBucket, and GitLab were the largest repositories by volume, there were a significant number of smaller repositories also discovered in the dataset. CodeCommit repositories, recently deprecated by AWS, were seen over 700 times. Approximately 3,500 of these smaller repositories were exposed during this operation. Many of these repositories are likely personal or being used by small groups.



Laravel Exploitation

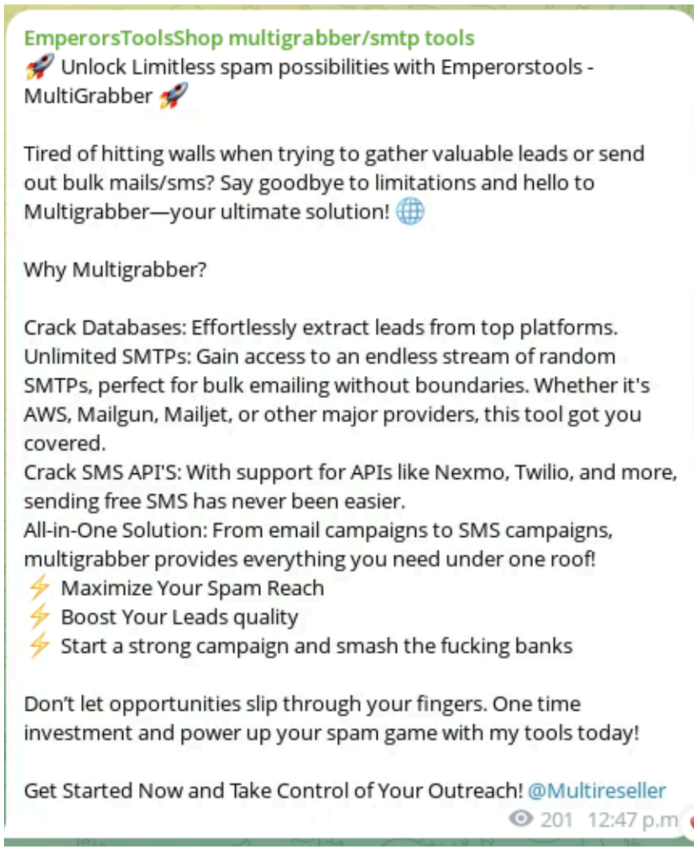
EMERALDWHALE, in addition to targeting Git configuration files, also targeted exposed Laravel environment files. Laravel, a PHP framework, has been a trendy choice for attackers in recent years and its vulnerabilities, targeting, and active exploitation have been widely reported on by CISA and Unit42. The *.env* files contain a wealth of credentials, including cloud service providers and databases.

The following diagram illustrates the attack path.


Website misconfigured
scan
.env exposed
extract
AWS creds Bucket S3 creds
check
Victim S3
Multigrabber v8.5 [running on vps/victim]
Store the results (AWS + other keys)

Multigrabber v8.5

There is an active market for Laravel exploitation tools and we will briefly present the one discovered throughout this research. Multigrabber is a secret-stealing tool that checks domains or IPs to validate if the .env file is present, and collects and classifies the information obtained to be used in spam or phishing campaigns. It is possible to find this tool in many forums and chats, and it has evolved in various versions adding new features. We found version 8.5 during the investigation. Here is an example of an advertisement in a Telegram group.



EmperorsToolsShop multigrabber/smtp tools
🚀 Unlock Limitless spam possibilities with Emperorstools - MultiGrabber 🚀

Tired of hitting walls when trying to gather valuable leads or send out bulk mails/sms? Say goodbye to limitations and hello to Multigrabber—your ultimate solution! 🌐

Why Multigrabber?

Crack Databases: Effortlessly extract leads from top platforms.
Unlimited SMTPs: Gain access to an endless stream of random SMTPs, perfect for bulk emailing without boundaries. Whether it's AWS, Mailgun, Mailjet, or other major providers, this tool got you covered.
Crack SMS API'S: With support for APIs like Nexmo, Twilio, and more, sending free SMS has never been easier.
All-in-One Solution: From email campaigns to SMS campaigns, multigrabber provides everything you need under one roof!
⚡ Maximize Your Spam Reach
⚡ Boost Your Leads quality
⚡ Start a strong campaign and smash the fucking banks

Don't let opportunities slip through your fingers. One time investment and power up your spam game with my tools today!

Get Started Now and Take Control of Your Outreach! @Multireseller
201 12:47 p.m

The official development team is EmperorsTool, but it seems to have stopped its activity. It appears that other groups with previous access to the code from EmperorsTool are now reselling.

Aftermath

The result of these attacks was over 15,000 credentials stolen for multiple different cloud services. We did not attempt to verify the validity of the credentials beyond basic regular expressions and simple deduplication. This attack was accomplished by just using scripts and exposed files on web servers, which led to another source of credentials: Git repositories.

Current Trends

Why are credential harvesting attacks becoming so common?

To answer this, we have monitored and detected over the last few months a multitude of attacks or automated scans in search of exposed files due to misconfiguration. Attackers are achieving their goals of stealing or obtaining credentials without much effort. In a nutshell:

- **Minimal effort**: Everything can be automated and they run their tools on temporary systems while saving the results elsewhere. It is becoming very difficult to know who is behind this kind of activity, which lowers the perceived risk to the attacker.
- **Free tools**: It is easy to find tools on GitHub that help with all of the necessary steps. There is also an active market for courses that would-be attackers can purchase.
- **Business**: It's fast income for the attackers. They confirm valid keys and sell them in packs or autoshops, websites, and Telegram bots that do not require any interaction.

Conclusion

EMERALDWHALE isn't the most sophisticated operation, but it still managed to collect over 15,000 credentials. It relied solely on misconfigurations rather than vulnerabilities, which isn't unique. What was different was the target: exposed Git configuration files. These files and the credentials they contain offer access to private repositories that normally would be difficult to access. In a private repository, developers may be more prone to include secrets because it offers a false sense of security.

The underground market for credentials is booming, especially for cloud services. This attack shows that secret management alone is not enough to secure an environment. There are just too many places credentials could leak from. **Monitoring the behavior of any identities associated with credentials is becoming a requirement to protect against these threats.**

Exposure Management and Vulnerability scanners can help in detecting issues, such as Git configuration files being viewable. It is important to also conduct these scans from both an internal and external perspective to get a full view of what attackers see.