

Tűzfal készítés mesterfokon: Zorp GPL 2.

1. Bevezető

a. Felvezetés az előző résztől

i. alapállapot előállítása

1. `ssh 172.16.12.202 -l adminla`
2. `sudo su -`
3. `cd /etc/zorp`
4. `vim instances.conf`
 - a. `default_instance --verbose=3 --policy /etc/zorp/policy.py`
5. `vim zones.py`
 - a. `from Zorp.Zone import Zone`
 - b.
 - c. `Zone('internet', addrs=['0.0.0.0/0', ':::0/0'])`
 - d.
 - e. `Zone(name='client', addrs=['192.168.200.0/24',],)`
 - f. `Zone(name='server', addrs=['192.168.201.0/24',],)`
6. `vim policy.py`
 - a. `from zones import *`
 - b.
 - c. `from Zorp.Core import *`
 - d.
 - e. `def default_instance():`
 - f. `pass`

2. SSL kapcsolat terminálása

a. https forgalom átengedése keybridge segítségévelvim

i. keybridge szabály felvétele

1. `vim policy.py`
- 2.
3. `from zones import *`
- 4.
5. `from Zorp.Core import *`
6. `from Zorp.Keybridge import X509KeyBridge`
7. `from Zorp.Pssl import *`
8. `from Zorp.Http import *`
- 9.
10. `class HttpProxyKeybridge(HttpProxy):`
11. `key_generator=X509KeyBridge(`
12. `key_file='/etc/zorp/keybridge/key.pem',`
13. `key_passphrase='passphrase',`
14. `cache_directory='/var/lib/zorp/keybridge-cache',`
15. `trusted_ca_files=(`
16. `'/etc/zorp/keybridge/ZorpGPL_TrustedCA.cert.pem',`
17. `'/etc/zorp/keybridge/ZorpGPL_TrustedCA.key.pem',`
18. `'passphrase'`
19. `),`
20. `untrusted_ca_files=(`
21. `'/etc/zorp/keybridge/ZorpGPL_UnTrustedCA.cert.pem',`
22. `'/etc/zorp/keybridge/ZorpGPL_UnTrustedCA.key.pem',`

```

23.     'passphrase'
24.     )
25.     )
26.
27.     def config(self):
28.         HttpProxy.config(self)
29.         self.require_host_header=False
30.         self.ssl.handshake_seq=SSL_HSO_SERVER_CLIENT
31.         self.ssl.client_keypair_generate=True
32.         self.ssl.client_connection_security=SSL_FORCE_SSL
33.         self.ssl.client_verify_type=SSL_VERIFY_OPTIONAL_UNTRUSTED
34.         self.ssl.server_connection_security=SSL_FORCE_SSL
35.         self.ssl.server_verify_type=SSL_VERIFY_REQUIRED_UNTRUSTED
36.         self.ssl.key_generator = self.key_generator
37.         self.ssl.server_ca_directory='/etc/ssl/certs'
38.         self.ssl.server_trusted_certs_directory='/etc/zorp/certs'
39.
40. def default_instance():
41.     Service(name='HttpsService',
42.         proxy_class=HttpsProxyKeybridge
43.     )
44.     Rule(service='HttpsService', dst_port=[443, ],
45.         src_zone=('client', ), dst_zone=('server', )
46.     )
47. mkdir keybridge
48. chown zorp:root keybridge
49. cd keybridge
50. wget https://raw.githubusercontent.com/balabit/zorp-
    examples/master/keybridge/ZorpGPL_TrustedCA.cert.pem
51. wget https://raw.githubusercontent.com/balabit/zorp-
    examples/master/keybridge/ZorpGPL\_TrustedCA.key.pem
52. wget https://raw.githubusercontent.com/balabit/zorp-
    examples/master/keybridge/ZorpGPL\_UnTrustedCA.cert.pem
53. wget https://raw.githubusercontent.com/balabit/zorp-
    examples/master/keybridge/ZorpGPL\_UnTrustedCA.key.pem
54. wget https://raw.githubusercontent.com/balabit/zorp-
    examples/master/keybridge/key.pem
55. cd ..

```

ii. keybridge szabály tesztelése

1. firewall

- a. service zorp reload
- b. tail -f /var/log/syslog

2. client

- a. w3m <https://192.168.201.254>
- b. openssl s_client -connect 192.168.201.254:443
- c. ssh 192.168.201.254 -p 443

b. smtps forgalom átengedése féloldalas ssl segítségével

i. kliens oldalon starttls szerver oldalon titkosítás nélkül (konfiguráció)

1. vim policy.py

```

2.
3. from zones import *
4.
5. from Zorp.Core import *
6. from Zorp.Pssl import *
7. from Zorp.Smtplib import *
8.
9. class SmtplibProxyStartTls(SmtplibProxy):
10.     def config(self):
11.         SmtplibProxy.config(self)
12.         self.relay_zones=('*',)
13.         self.ssl.client_connection_security = SSL_ACCEPT_STARTTLS
14.         self.ssl.client_verify_type = SSL_VERIFY_OPTIONAL_UNTRUSTED
15.         self.ssl.client_keypair_files=(
16.             '/etc/ssl/certs/ssl-cert-snakeoil.pem',
17.             '/etc/ssl/private/ssl-cert-snakeoil.key'
18.         )
19.         self.ssl.server_verify_type = SSL_VERIFY_OPTIONAL_UNTRUSTED
20.
21. def default_instance():
22.     Service(name='service_smtplib_transparent_starttls',
23.             proxy_class=SmtplibProxyStartTls
24.             )
25.
26.     Rule(service='service_smtplib_transparent_starttls', dst_port=25,
27.          src_zone=('client'), dst_zone=('server'))
28.     )

```

ii. kliens oldalon starttls szerver oldalon titkosítás nélkül (tesztelés)

1. firewall

a. service zorp reload

2. client

a. echo -e 'EHLO client\r\nQUIT\r\n' | nc 192.168.201.254 25

- i. 220 server ESMTP Postfix (Debian/GNU)
- ii. 250-server
- iii. 250-PIPELINING
- iv. 250-SIZE 10240000
- v. 250-ETRN
- vi. 250-STARTTLS
- vii. 250 8BITMIME
- viii. 221 2.0.0 Bye

iii. kliens oldalon titkosítás nélkül szerver oldalon ssl (konfiguráció)

```

1. vim policy.py
2.
3. from zones import *
4.
5. from Zorp.Core import *
6. from Zorp.Pssl import *
7. from Zorp.Smtplib import *
8.

```

```

9. class SmtProxyOneSideSsl(SmtProxy):
10.     def config(self):
11.         SmtProxy.config(self)
12.         self.relay_zones=('*',)
13.         self.ssl.server_connection_security=SSL_FORCE_SSL
14.         self.ssl.server_verify_type=SSL_VERIFY_OPTIONAL_UNTRUSTED
15.
16. def default_instance():
17.     Service(name='service_smtproxy_transparent_one_sided_ssl',
18.             proxy_class=SmtProxyOneSideSsl,
19.             router=DirectedRouter(dest_addr=(SockAddrInet('192.168.201.254',
20. 465),))
21.     )
22.     Rule(service='service_smtproxy_transparent_one_sided_ssl', dst_port=25,
23.          src_zone=('client'), dst_zone=('server'))
24.     )

```

iv.kliens oldalon titkosítás nélkül szerver oldalon ssl (tesztelés)

1. *firewall*
 - a. service zorp reload
 - b. grc tail -n 0 -f /var/log/syslog
2. *client*
 - a. echo -e 'EHLO client\r\nQUIT\r\n' | nc 192.168.201.254 25
 - i. 220 server ESMTP Postfix (Debian/GNU)
 - ii.250-server
 - iii.250-PIPELINING
 - iv.250-SIZE 10240000
 - v.250-ETRN
 - vi.250 8BITMIME
 - vii.221 2.0.0 Bye

3. Audit

a. Applikációs szintű naplózás

i. audit szabályok hozzáadása

```

1. vim instances.conf
2.
3. default_instance --verbose=3 --policy /etc/zorp/policy.py
4.
5. vim policy.py
6.
7. from zones import *
8.
9. from Zorp.Core import *
10. from Zorp.Ftp import *
11. from Zorp.Http import *
12.
13. def default_instance():
14.     Service(name='service_ftp_transparent_audit', proxy_class=FtpProxy
15.             )
16.     Service(name='service_http_transparent_audit', proxy_class=HttpProxy

```

```

17.     )
18.
19.     Rule(service='service_ftp_transparent_audit', dst_port=21,
20.         src_zone=('client', ), dst_zone=('server', )
21.     )
22.     Rule(service='service_http_transparent_audit', dst_port=80,
23.         src_zone=('client', ), dst_zone=('server', )
24.     )
25.

```

ii. http szabály tesztelése (logspec nélkül)

1. *firewall*
 - a. service zorp restart
 - b. grc tail -n 0 -f /var/log/syslog
2. *client*
 - a. w3m <http://192.168.201.254>
 - b. w3m ftp://adminla@192.168.201.254

iii.audit logspec hozzáadása

1. vim instances.conf
- 2.
3. default_instance --verbose=3 --logspec
'http.accounting:4,ftp.request:6,ftp.response:6,pop3.request:7,pop3.response:7,
smtp.request:7,smtp.response:7' --policy /etc/zorp/policy.py

iv.audit logspec tesztelése

1. *firewall*
 - a. service zorp restart
 - b. grc tail -n 0 -f /var/log/syslog
2. *client*
 - a. w3m <http://192.168.201.254>
 - b. w3m ftp://adminla@192.168.201.254

4. Header manipulálása

a. Header replace

i. http szabály hozzáadása

```

1. vim policy.py
2.
3. from zones import *
4.
5. from Zorp.Core import *
6. from Zorp.Http import *
7.
8. def default_instance():
9.     Service(name='service_http_transparent',
10.         proxy_class=HttpProxy
11.     )
12.
13.     Rule(service='service_http_transparent', dst_port=80,
14.         src_zone=('client', ), dst_zone=('server', )
15.     )
16.

```

ii. http szabály tesztelése

1. *firewall*
 - a. `service zorp reload`
 - b. `grc tail -n 0 -f /var/log/syslog`
2. *server*
 - a. `tail -n 0 -f /var/log/apache2/access.log`
3. *client*
 - a. `w3m http://192.168.201.254`

iii. **http header replace szabály hozzáadása**

1. `vim policy.py`
- 2.
3. `from zones import *`
- 4.
5. `from Zorp.Core import *`
6. `from Zorp.Http import *`
- 7.
8. `class HttpProxyHeaderReplace(HttpProxy):`
9. `def config(self):`
10. `HttpProxy.config(self)`
11. `self.request_header["User-Agent"] = (HTTP_HDR_CHANGE_VALUE,`
`"Forged Browser 1.0")`
- 12.
13. `def default_instance():`
14. `Service(name="service_http_transparent_header_replace",`
15. `proxy_class=HttpProxyHeaderReplace`
16. `)`
- 17.
18. `Rule(service='service_http_transparent_header_replace', dst_port=80,`
19. `src_zone=('client',), dst_zone=('server',)`
20. `)`

iv. **http header replace szabály tesztelése**

1. *firewall*
 - a. `service zorp reload`
 - b. `grc tail -n 0 -f /var/log/syslog`
2. *server*
 - a. `tail -n 0 -f /var/log/apache2/access.log`
3. *client*
 - a. `w3m http://192.168.201.254`

5. **Tartalom manipulálása**

a. **Tartalom stackelt proxyval**

i. **http content passthrough anypy proxyval (konfiguráció)**

1. `vim policy.py`
- 2.
3. `from zones import *`
- 4.
5. `from Zorp.Core import *`
6. `from Zorp.AnyPy import *`
7. `from Zorp.Http import *`
8. `from Zorp.Stream import *`
- 9.

```

10. class AnyPyProxyCat(AnyPyProxy):
11.     def config(self):
12.         AnyPyProxy.config(self)
13.
14.     def proxyThread(self):
15.         while True:
16.             try:
17.                 line=self.client_stream.readline()
18.             except StreamException, (code, line):
19.                 if code == G_IO_STATUS_EOF:
20.                     return
21.                 else:
22.                     raise
23.             self.server_stream.write(line)
24.
25.
26. class HttpProxyStackAnyPy(HttpProxy):
27.     def config(self):
28.         HttpProxy.config(self)
29.
30.     self.request_header['Accept-
    Encoding']=(HTTP_HDR_CHANGE_VALUE, 'identity')
31.     self.response_stack['GET'] = (HTTP_STK_DATA, (Z_STACK_PROXY,
    AnyPyProxyCat))
32.
33. def default_instance():
34.     Service(name='service_http_transparent_stack_cat',
35.         proxy_class=HttpProxyStackAnyPy
36.     )
37.     Rule(service='service_http_transparent_stack_cat', dst_port=80,
38.         src_zone=('client', ), dst_zone=('server', )
39.     )
40.

```

ii. http content passthrough anypy proxyval (tesztelés)

1. firewall

- a. service zorp reload
- b. grc tail -n 0 -f /var/log/syslog

2. server

- a. tail -n 0 -f /var/log/apache2/access.log

3. client

- a. w3m <http://192.168.201.254>

b. Tartalom másolás külső programmal

i. http content passthrough külső programmal (konfiguráció)

```

1. vim policy.py
2.
3. from zones import *
4.
5. from Zorp.Core import *
6. from Zorp.Http import *

```

```

7.
8. class HttpProxyStackCat(HttpProxy):
9.     def config(self):
10.         HttpProxy.config(self)
11.         self.response_stack['GET'] = (HTTP_STK_DATA,
12.             (Z_STACK_PROGRAM, '/bin/cat'))
13.
14. def default_instance():
15.     Service(name='service_http_transparent_stack_cat',
16.         proxy_class=HttpProxyStackCat
17.     )
18.     Rule(service='service_http_transparent_stack_cat', dst_port=80,
19.         src_zone=('client', ), dst_zone=('server', )
20.     )

```

ii. http content passthrough külső programmal (tesztelés)

1. firewall

- a. service zorp reload
- b. grc tail -n 0 -f /var/log/syslog

2. client

- a. w3m <http://192.168.201.254>

c. Tartalom manipulálása helyben

i. http content uppercase stackelt proxyval (konfiguráció)

```

1. vim policy.py
2.
3. from zones import *
4.
5. from Zorp.Core import *
6. from Zorp.AnyPy import *
7. from Zorp.Http import *
8. from Zorp.Stream import *
9.
10. class AnyPyProxyTr(AnyPyProxy):
11.     def config(self):
12.         AnyPyProxy.config(self)
13.
14.     def proxyThread(self):
15.         while True:
16.             try:
17.                 line=self.client_stream.readline()
18.             except StreamException, (code, line.upper()):
19.                 if code == G_IO_STATUS_EOF:
20.                     return
21.                 else:
22.                     raise
23.             self.server_stream.write(line.upper())
24.
25.
26. class HttpProxyStackAnyPy(HttpProxy):
27.     def config(self):

```



```

28.         HttpProxy.config(self)
29.
30.         self.request_header['Accept-
Encoding']=(HTTP_HDR_CHANGE_VALUE, 'identity')
31.         self.response_stack['GET'] = (HTTP_STK_DATA, (Z_STACK_PROXY,
AnyPyProxyTr))
32.
33. def default_instance():
34.     Service(name='service_http_transparent_stack_cat',
35.         proxy_class=HttpProxyStackAnyPy
36.     )
37.     Rule(service='service_http_transparent_stack_cat', dst_port=80,
38.         src_zone=('client', ), dst_zone=('server', )
39.     )
40.
41.

```

ii. http content uppercase stackelt proxyval (tesztelés)

1. *firewall*
 - a. service zorp reload
 - b. grc tail -n 0 -f /var/log/syslog
2. *client*
 - a. w3m <http://192.168.201.254>

d. Tartalom manipulálása külső programmal

i. http content uppercase külső programmal (konfiguráció)

```

1. vim policy.py
2.
3. from zones import *
4.
5. from Zorp.Core import *
6. from Zorp.Http import *
7.
8. class HttpProxyStackTr(HttpProxy):
9.     def config(self):
10.         HttpProxy.config(self)
11.         self.request_header['Accept-Encoding'] = (HTTP_HDR_POLICY,
self.processAcceptEncoding)
12.         self.response_stack['GET'] = (HTTP_STK_DATA,
(Z_STACK_PROGRAM, '/usr/bin/tr '[a-z]' '[A-Z]'))
13.
14.     def processAcceptEncoding(self, name, value):
15.         lst_value = value.split(',')
16.         if 'gzip' in lst_value:
17.             lst_value.remove('gzip')
18.         if 'bzip' in lst_value:
19.             lst_value.remove('bzip')
20.         if 'bzip2' in lst_value:
21.             lst_value.remove('bzip2')
22.         if 'compress' in lst_value:
23.             lst_value.remove('compress')

```

```

24.         self.current_header_value = ','.join(lst_value)
25.
26.         return HTTP_HDR_ACCEPT
27.
28. def default_instance():
29.     Service(name='service_http_transparent_stack_tr',
30.           proxy_class=HttpProxyStackTr
31.           )
32.     Rule(service='service_http_transparent_stack_tr', dst_port=80,
33.          src_zone=('client', ), dst_zone=('server', )
34.          )

```

ii. http content uppercase külső programmal (konfiguráció)

1. firewall

- a. service zorp reload
- b. grc tail -n 0 -f /var/log/syslog

2. client

- a. w3m <http://192.168.201.254>

6. Tartalom ellenőrzése

a. Tartalom ellenőrzése helyben

i. Minimális URL filter (konfiguráció)

```

1. vim policy.py
2.
3. from zones import *
4.
5. from Zorp.Core import *
6. from Zorp.Http import *
7.
8. class HttpProxyUrlFilter(HttpProxy):
9.     def config(self):
10.         HttpProxy.config(self)
11.         self.request['GET'] = (HTTP_REQ_POLICY, self.filterURL)
12.
13.     def filterURL(self, method, url, version):
14.         if (url == 'http://192.168.201.254/disallowed_content.html'):
15.             self.error_info = 'Access of this content is denied by the local
policy.'
16.             return HTTP_REQ_REJECT
17.             return HTTP_REQ_ACCEPT
18.
19. def default_instance():
20.     Service(name='service_http_transparent_url_filter',
21.           proxy_class=HttpProxyUrlFilter
22.           )
23.     Rule(service='service_http_transparent_url_filter', dst_port=80,
24.          src_zone=('client', ), dst_zone=('server', )
25.          )

```

ii. Minimális URL filter (tesztelés)

1. firewall

- a. service zorp reload

- b. `grc tail -n 0 -f /var/log/syslog`
 2. *server*
 - a. `tail -n 0 -f /var/log/apache2/access.log`
 3. *client*
 - a. `w3m http://192.168.201.254/disallowed_content.html`
- b. **Tartalom ellenőrzése külső programmal**
 - i. **Vírusellenőrzés külső programmal (konfiguráció)**
 1. `vim policy.py`
 - 2.
 3. `from zones import *`
 - 4.
 5. `from Zorp.Core import *`
 6. `from Zorp.Ftp import *`
 7. `from Zorp.Http import *`
 - 8.
 9. `class HttpProxyStackClamav(HttpProxy):`
 10. `def config(self):`
 11. `HttpProxy.config(self)`
 12. `self.keep_persistent = TRUE`
 13. `self.response_stack['GET'] = (HTTP_STK_DATA,`
`(Z_STACK_PROGRAM, '/etc/zorp/scripts/clamav_stack.py'))`
 - 14.
 15. `class FtpProxyStackClamav(FtpProxy):`
 16. `def config(self):`
 17. `FtpProxy.config(self)`
 18. `self.request_stack['RETR']=(FTP_STK_DATA, (Z_STACK_PROGRAM,`
`'/etc/zorp/scripts/clamav_stack.py'))`
 - 19.
 20. `def default_instance():`
 21. `Service(name='service_http_transparent_stack_clamav',`
 22. `proxy_class=HttpProxyStackClamav`
 23. `)`
 24. `Service(name='service_ftp_transparent_stack_clamav',`
 25. `proxy_class=FtpProxyStackClamav`
 26. `)`
 - 27.
 28. `Rule(service='service_http_transparent_stack_clamav', dst_port=80,`
 29. `src_zone=('client',), dst_zone=('server',)`
 30. `)`
 31. `Rule(service='service_ftp_transparent_stack_clamav', dst_port=21,`
 32. `src_zone=('client',), dst_zone=('server',)`
 33. `)`
 - 34.
 35. `mkdir scripts`
 36. `vim scripts/clamav_stack.py`
 - 37.
 38. `#!/usr/bin/python -B`
 - 39.
 40. `import os`

```

41. import syslog
42. import tempfile
43. import pyclamd
44.
45. def copy_file(fd_in, fd_out):
46.     while True:
47.         copy_buffer = os.read(fd_in, 1024)
48.         if copy_buffer:
49.             os.write(fd_out, copy_buffer)
50.         else:
51.             break
52. def scan_input():
53.     try:
54.         (tmp_file, tmp_file_name) = tempfile.mkstemp()
55.         os.fchmod(tmp_file, 0644)
56.         copy_file(0, tmp_file)
57.         os.close(0)
58.     except OSError as e:
59.         syslog.syslog('Temporary file creation failed: \'%s\'\'\' % str(e))
60.         return (None, 'none', {'none' : 'scan failed'})
61.
62.     try:
63.         pyclamd.init_unix_socket()
64.         found_virus = pyclamd.scan_file(tmp_file_name)
65.     except pyclamd.ScanError as e:
66.         syslog.syslog('Virus scan failed: \'%s\'\'\' % str(e))
67.         return (tmp_file, tmp_file_name, {tmp_file_name : 'scan failed'})
68.
69.     return (tmp_file, tmp_file_name, found_virus)
70.
71. def main():
72.     syslog.openlog('clamav_stack')
73.
74.     (tmp_file, tmp_file_name, found_virus) = scan_input()
75.     if found_virus or tmp_file == None:
76.         try:
77.             syslog.syslog('%s in the content.\n' % found_virus[tmp_file_name])
78.
79.             fd = os.fdopen(3, 'w')
80.             fd.write('0 SETVERDICT\n[]Verdict\nZ_REJECT\n[]Description\n%s in
the content.\n\n' % found_virus[tmp_file_name])
81.             fd.close()
82.         except OSError as e:
83.             syslog.syslog('Set verdict failed: \'%s\'\'\' % str(e))
84.         else:
85.             try:
86.                 syslog.syslog('No viruses were found in the content.\n')
87.                 os.lseek(tmp_file, 0, os.SEEK_SET)
88.                 copy_file(tmp_file, 1)

```

```

89.         except OSError as e:
90.             syslog.syslog('No viruses were found in the content.\n')
91.
92.         os.close(1)
93.         if tmp_file:
94.             os.close(tmp_file)
95.             os.unlink(tmp_file_name)
96.
97. if __name__ == '__main__':
98.     main()
99.
100.chmod +x scripts/clamav_stack.py

```

ii. Vírusellenőrzés külső programmal (tesztelés)

1. *firewall*

- a. service zorp reload
- b. grc tail -n 0 -f /var/log/syslog

2. *server*

- a. mkdir public_html
- b. cd public_html
- c. vim index.html
 - i. It Works!
- d. wget <http://www.eicar.org/download/eicar.com>
- e. wget <http://www.eicar.org/download/eicar.com.txt>
- f. wget <http://www.eicar.org/download/eicar.com.zip>
- g. wget <http://www.eicar.org/download/eicarcom2.zip>
- h. cd -
- i. tail -n 0 -f /var/log/clamav/clamav.log

3. *client*

- a. w3m <http://192.168.201.254/~adminla/eicar.com>
- b. w3m <http://192.168.201.254/~adminla/eicar.com.txt>
- c. w3m <http://192.168.201.254/~adminla/eicar.com.zip>
- d. w3m <http://192.168.201.254/~adminla/eicarcom2.zip>

7. Komplex példa

a. Google Calendar titkosítása

i. Google Calendar titkosítás (konfiguráció)

```

1. vim policy.py
2.
3. from Zorp.Core import *
4. from Zorp.Pssl import *
5. from Zorp.Http import *
6. from Zorp.AnyPy import *
7. from Zorp.Proxy import *
8.
9. import Zorp
10. from Zorp import Stream
11.
12. from zones import *
13.
14. import DataHandler

```

```

15. import Cypher
16.
17. config.options.kzorp_enabled=False
18.
19. class CloudEncryptionStackedProxy(AnyPyProxy):
20.     def config(self):
21.         self.client_max_line_length = 65535
22.
23.         self.data_handler = DataHandler.GoogleCalendarProxyDataHandler()
24.         self.cypher = Cypher.Base64Cypher("")
25.
26.     def readData(self):
27.         data = ""
28.         while True:
29.             try:
30.                 data += self.client_stream.read(1024)
31.             except Stream.StreamException, (code, lastline):
32.                 if code == Stream.G_IO_STATUS_EOF:
33.                     break
34.                 else:
35.                     raise
36.
37.         return data
38.
39.     def proxyThread(self):
40.         try:
41.             plain_raw_data = self.readData()
42.             self.data_handler.parse(plain_raw_data)
43.             self.data_handler.cypher(self.cypher, self.is_request)
44.             cyphered_raw_data = self.data_handler.compose()
45.             self.server_stream.write(cyphered_raw_data)
46.             #proxyLog(self, "cypher.info", 3, "Data successfully cyphered")
47.         except KeyError as e:
48.             proxyLog(self, "cypher.error", 3, "Data cypher failed; error='%s'", str(e))
49.
50.
51. class CloudEncryptionStackedProxyRequest(CloudEncryptionStackedProxy):
52.     def config(self):
53.         super(self.__class__, self).config()
54.
55.         self.is_request = True
56.
57.
58. class CloudEncryptionStackedProxyResponse(CloudEncryptionStackedProxy):
59.     def config(self):
60.         super(self.__class__, self).config()
61.
62.         self.is_request = False
63.

```

```

64.
65. class CloudEncryptionSSLProxy(HttpProxy):
66.     def config(self):
67.         HttpProxy.config(self)
68.
69.         self.transparent_mode=TRUE
70.
71.         self.ssl.handshake_seq=SSL_HSO_SERVER_CLIENT
72.         self.ssl.key_generator=X509KeyBridge(
73.             key_file="/etc/zorp/keybridge/key.pem",
74.             key_passphrase="passphrase",
75.             cache_directory="/var/lib/zorp/keybridge-cache",
76.             trusted_ca_files=(
77.                 "/etc/zorp/keybridge/ZorpGPL_TrustedCA.cert.pem",
78.                 "/etc/zorp/keybridge/ZorpGPL_TrustedCA.key.pem",
79.                 "passphrase"
80.             ),
81.             untrusted_ca_files=(
82.                 "/etc/zorp/keybridge/ZorpGPL_UnTrustedCA.cert.pem",
83.                 "/etc/zorp/keybridge/ZorpGPL_UnTrustedCA.key.pem",
84.                 "passphrase"
85.             )
86.         )
87.
88.         self.ssl.client_connection_security=SSL_FORCE_SSL
89.         self.ssl.client_keypair_generate=TRUE
90.         self.ssl.client_ssl_method=SSL_METHOD_ALL
91.         self.ssl.client_disable_proto_sslv2=TRUE
92.
93.         self.ssl.server_connection_security=SSL_FORCE_SSL
94.         self.ssl.server_verify_depth=5
95.         self.ssl.server_verify_type=SSL_VERIFY_NONE
96.         self.ssl.client_verify_type=SSL_VERIFY_NONE
97.         self.ssl.server_check_subject=FALSE
98.
99.
100. class CloudEncryptionHttpsProxy(CloudEncryptionSSLProxy):
101.     def config(self):
102.         CloudEncryptionSSLProxy.config(self)
103.
104.         self.request_header["Accept-
            Encoding"]=(HTTP_HDR_CHANGE_VALUE, "identity")
105.         self.rewrite_host_header = FALSE
106.
107.         self.request["GET"] = (HTTP_REQ_POLICY,
            self.filterOutIrrelevantTraffic)
108.         self.request["POST"] = (HTTP_REQ_POLICY,
            self.filterOutIrrelevantTraffic)
109.

```

```

110.     def filterOutIrrelevantTraffic(self, method, url, version):
111.         raise NotImplementedError
112.
113.
114. class
    CloudEncryptionHttpsGoogleCalendarProxy(CloudEncryptionHttpsProxy):
115.     def filterOutIrrelevantTraffic(self, method, url, version):
116.         proxyLog(self, "", 3, "Data cypher failed; uri='%s'", self.request_url_file)
117.         if self.request_url_file.startswith("/calendar/feeds"):
118.             self.request_stack["*"] = (HTTP_STK_DATA, (Z_STACK_PROXY,
                CloudEncryptionStackedProxyRequest))
119.             self.response_stack["*"] = (HTTP_STK_DATA, (Z_STACK_PROXY,
                CloudEncryptionStackedProxyResponse))
120.             return HTTP_REQ_ACCEPT
121.
122.
123. class CloudEncryptionHttpNonTransparentProxy(HttpProxyNonTransparent):
124.     def config(self):
125.         HttpProxyNonTransparent.config(self)
126.         self.connect_proxy=CloudEncryptionHttpsGoogleCalendarProxy
127.         self.request["*"]=HTTP_REQ_ACCEPT
128.
129.
130. def default_instance():
131.     Service("cloud_encryption_service",
        CloudEncryptionHttpNonTransparentProxy, router=InbandRouter())
132.     Listener(SocketAddrInet("172.16.12.202", 8080),
        "cloud_encryption_service")
133.
134.
135. vim DataHandler.py
136.
137. from syslog import syslog
138.
139. class ProxyDataHandlerBase(object):
140.     STATE_INIT = 0
141.     STATE_PARSED = 1
142.     STATE_CYPHERED = 2
143.
144.     def __init__(self):
145.         self._state = ProxyDataHandlerBase.STATE_INIT
146.
147.     def parse(self, plain_raw_data):
148.         self._state = ProxyDataHandlerBase.STATE_PARSED
149.
150.         self._parse(plain_raw_data)
151.
152.     def cypher(self, cypher, is_request):
153.         if self._state < ProxyDataHandlerBase.STATE_PARSED:

```



```

154.         raise TypeError('no plain data to cypher')
155.
156.         syslog.syslog('%s %s' % (str(is_request), 'alma'))
157.
158.         cypher_func = cypher.encrypt if is_request else cypher.decrypt
159.         self._cypher(cypher_func)
160.
161.         self._state = ProxyDataHandlerBase.STATE_CYPHERED
162.
163.         def compose(self):
164.             if self._state < ProxyDataHandlerBase.STATE_CYPHERED:
165.                 raise TypeError('no chypered data to compose')
166.
167.             return self._compose()
168.
169.         def log(self, logger):
170.             pass
171.
172.         def __dict__(self):
173.             if self._state < ProxyDataHandlerBase.STATE_PARSED:
174.                 raise TypeError('no chypered data')
175.
176.             return {}
177.
178.
179. import xml.etree.ElementTree as ET
180. class XMLProxyDataHandler(ProxyDataHandlerBase):
181.     def _parse(self, plain_raw_data):
182.         syslog("%s" % plain_raw_data)
183.         try:
184.             self._parsed_data = ET.fromstring(plain_raw_data)
185.         except ET.ParseError as e:
186.             raise e
187.
188.     def _compose(self):
189.         try:
190.             cyphered_raw_data = ET.tostring(self._parsed_data)
191.             return cyphered_raw_data
192.         except AttributeError as e:
193.             raise TypeError(e)
194.
195. import gdata
196. import syslog
197. import xml.etree.ElementTree as ET
198. import xml.etree.cElementTree as cET
199. class GoogleCalendarProxyDataHandler(ProxyDataHandlerBase):
200.     namespaces = { "": 'http://www.w3.org/2005/Atom',
201.                    'gCal': 'http://schemas.google.com/gCal/2005',
202.                    'gd': 'http://schemas.google.com/g/2005',

```

```

203.         'openSearch' : 'http://a9.com/-/spec/opensearchrss/1.0/',
204.     }
205.     namespaces_registered = False
206.
207.     def __init__(self):
208.         if not GoogleCalendarProxyDataHandler.namespaces_registered:
209.             for (prefix, uri) in
                GoogleCalendarProxyDataHandler.namespaces.iteritems():
210.                 ET.register_namespace(prefix, uri)
211.             GoogleCalendarProxyDataHandler.namespaces_registered = True
212.
213.         def _parse(self, plain_raw_data):
214.             syslog.syslog('parse \'%s\' % plain_raw_data)
215.             try:
216.                 self.gdata = gdata.GDataEntryFromString(plain_raw_data)
217.                 if self.gdata == None:
218.                     raise cET.ParseError
219.                 syslog.syslog('parsed as gdataentry %s' % str(type(self.gdata)))
220.                 return
221.             except cET.ParseError:
222.                 pass
223.             try:
224.                 self.gdata = gdata.GDataFeedFromString(plain_raw_data)
225.                 if self.gdata == None:
226.                     raise cET.ParseError
227.                 syslog.syslog('parsed as gdata %s' % str(type(self.gdata)))
228.                 return
229.             except cET.ParseError:
230.                 pass
231.             self.gdata = None
232.             syslog.syslog('unknown %s' % plain_raw_data)
233.
234.         def _cypher(self, cypher_func):
235.             if isinstance(self.gdata, gdata.GDataFeed):
236.                 entries = self.gdata.entry
237.             elif isinstance(self.gdata, gdata.GDataEntry):
238.                 entries = [self.gdata, ]
239.             else:
240.                 return
241.
242.             for entry in entries:
243.                 entry.title.text = cypher_func(entry.title.text)
244.
245.         def _compose(self):
246.             return str(self.gdata)
247.
248.vim Cypher.py
249.
250.class BaseCypher(object):

```

```

251.     def __init__(self, magic=""):
252.         self.magic = magic
253.         self.magic_len = len(magic)
254.
255.     def encrypt(self, plaintext):
256.         cyphertext = self.magic + self._encrypt(plaintext)
257.         return cyphertext
258.
259.     def _encrypt(self, plaintext):
260.         raise NotImplementedError
261.
262.     def decrypt(self, cyphertext):
263.         if self.magic_len > 0 and len(cyphertext) < self.magic_len:
264.             raise IndexError
265.         plaintext = self._decrypt(cyphertext[self.magic_len:])
266.         return plaintext
267.
268.     def _decrypt(self, cyphertext):
269.         raise NotImplementedError
270.
271. class NoCypher(BaseCypher):
272.     def __init__(self, magic=""):
273.         super(NoCypher, self).__init__(magic)
274.
275.     def _encrypt(self, plaintext):
276.         return plaintext
277.
278.     def _decrypt(self, cyphertext):
279.         return cyphertext
280.
281.
282. class Base64Cypher(BaseCypher):
283.     base64 = __import__('base64')
284.
285.     def __init__(self, magic=""):
286.         super(Base64Cypher, self).__init__(magic)
287.
288.     def _encrypt(self, plaintext):
289.         return Base64Cypher.base64.b64encode(plaintext)
290.
291.     def _decrypt(self, cyphertext):
292.         return Base64Cypher.base64.b64decode(cyphertext)

```

ii. Google Calendar titkosítás (tesztelés)

1. *firewall*

- a. service zorp reload
- b. grc tail -n 0 -f /var/log/syslog

2. *client*

- a. Thunderbird HTTP proxy beállítása
- b. Event felvétele

- c. Event megtekintése
- d. Firefox átváltás
 - i. <http://calendar.google.com>
 - ii. zorptest@gmail.com
- e. Event megtekintése
- f. `echo "event field value" | base64 -d`