

Hunter Donald

hxd0011

COMP 5370

Homework 3

1)

- a. The limitations and capabilities of access control mechanisms differ in distributed systems because they may use a variety of access control mechanisms that must be integrated to support the organization's policy.
  - i. Lightweight Directory Access Protocol allows a modular, expandable access control and single sign-on solution to be deployed rapidly for all applications in the information system.
  - ii. Kerberos uses a trusted third party which provides a means by which constituents of the network can trust each other. These constituents may be any hardware or software that communicates across the network.
- b. "chmod u+rwx, g+rx, o-rwx Auburn\_Security\_Mechanisms"

2)

```
File Edit View Search Terminal Help
root@kali:~# gpg --full-gen-key
gpg (GnuPG) 2.2.17; Copyright (C) 2019 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
 (1) RSA and RSA (default)
 (2) DSA and Elgamal
 (3) DSA (sign only)
 (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n> = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0) 1
Key expires at Thu 24 Oct 2019 09:02:23 PM EDT
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: hunterdonald
Email address: hxd0011@auburn.edu
Comment: COMP5370
You selected this USER-ID:
    "hunterdonald (COMP5370) <hxd0011@auburn.edu>"

Change (N)ame, (C)omment, (E)mail or (O)key/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key A92AFA72100C2282 marked as ultimately trusted
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/87D6D7385367E1414E4996F2A92AFA72100C2282.rev'
public and secret key created and signed.
```

Hunter Donald

hzd0011

COMP 5370

Homework 3

```
root@kali:~
```

File Edit View Search Terminal Help

```
public and secret key created and signed.
```

```
pub rsa2048 2019-10-24 [SC] [expires: 2019-10-25]
87D6D7385367E1414E4996F2A92FA72100C2282
uid [ultimate] hunterdonald (COMP5370) <hzd0011@auburn.edu>
sub rsa2048 2019-10-24 [E] [expires: 2019-10-25]
```

```
root@kali:~# gpg --list-secret-keys
gpg: checking the trustdb
gpg: marginal needed: 3 completes needed: 1  trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, lu
gpg: next trustdb check due at 2019-10-25
/root/.gnupg/pubring.kbx
```

```
sec rsa2048 2019-10-24 [SC] [expires: 2019-10-25]
87D6D7385367E1414E4996F2A92FA72100C2282
uid [ultimate] hunterdonald (COMP5370) <hzd0011@auburn.edu>
ssb rsa2048 2019-10-24 [E] [expires: 2019-10-25]
```

```
root@kali:~# gpg --list-keys
/root/.gnupg/pubring.kbx
```

```
pub rsa2048 2019-10-24 [SC] [expires: 2019-10-25]
87D6D7385367E1414E4996F2A92FA72100C2282
uid [ultimate] hunterdonald (COMP5370) <hzd0011@auburn.edu>
sub rsa2048 2019-10-24 [E] [expires: 2019-10-25]
```

```
root@kali:~# gpg --export --output hunterdonald.gpg hunterdonald
root@kali:~# apache2ctl start
Invoking 'systemctl start apache2'.
Use 'systemctl status apache2' for more info.
root@kali:~# cp hunterdonald.gpg /var/www/html
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.6 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::a00:27ff:fe7c:8e8e prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:7c:8e:8e txqueuelen 1000 (Ethernet)
                RX packets 214 bytes 43045 (42.0 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 30 bytes 2813 (2.7 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
```

```
root@kali:~
```

File Edit View Search Terminal Help

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.6 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::a00:27ff:fe7c:8e8e prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:7c:8e:8e txqueuelen 1000 (Ethernet)
                RX packets 214 bytes 43045 (42.0 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 30 bytes 2813 (2.7 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
                RX packets 20 bytes 1116 (1.0 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 20 bytes 1116 (1.0 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
root@kali:~# nc -l 4444 > test.txt
^C
root@kali:~# ls -ll
total 1160
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Desktop
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Documents
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Downloads
-rw-r--r-- 1 root root 207 Sep 27 12:07 hunterDonald-EssentialSoftwarePatch.elf
-rw-r--r-- 1 root root 1252 Oct 23 21:04 hunterdonald.gpg
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Music
-rw-r--r-- 1 root root 1146630 Sep 27 14:10 packets.pcap
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Pictures
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Public
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Templates
-rw-r--r-- 1 root root 0 Oct 23 21:15 test.txt
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Videos
root@kali:~# nc -l -p 4444 > test.txt
root@kali:~# ls -ll
total 1164
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Desktop
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Documents
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Downloads
-rw-r--r-- 1 root root 207 Sep 27 12:07 hunterDonald-EssentialSoftwarePatch.elf
-rw-r--r-- 1 root root 1252 Oct 23 21:04 hunterdonald.gpg
drwxr-xr-x 2 root root 4096 Sep  9 10:09 Music
-rw-r--r-- 1 root root 1146630 Sep 27 14:10 packets.pcap
```

Hunter Donald

hxd0011

COMP 5370

Homework 3

```
root@kali:~
```

File Edit View Search Terminal Help

```
-rw-r--r-- 1 root root 207 Sep 27 12:07 hunterDonald-EssentialSoftwarePatch.elf
-rw-r--r-- 1 root root 1252 Oct 23 21:04 hunterdonald.gpg
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Music
-rw-r--r-- 1 root root 1146630 Sep 27 14:10 packets.pcap
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Pictures
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Public
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Templates
-rw-r--r-- 1 root root 0 Oct 23 21:15 test.txt
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Videos
root@kali:~# nc -l -p 4444 > test.txt
root@kali:~# ls -ll
total 1164
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Desktop
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Documents
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Downloads
-rw-r--r-- 1 root root 207 Sep 27 12:07 hunterDonald-EssentialSoftwarePatch.elf
-rw-r--r-- 1 root root 1252 Oct 23 21:04 hunterdonald.gpg
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Music
-rw-r--r-- 1 root root 1146630 Sep 27 14:10 packets.pcap
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Pictures
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Public
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Templates
-rw-r--r-- 1 root root 382 Oct 23 21:18 test.txt
drwxr-xr-x 2 root root 4096 Sep 9 10:09 Videos
root@kali:~# cat test.txt
0
00h00x001a"qn000(000|0500+L0000>0cc000 $[000H&0w000:0]0R00FB0R0H[0_r00i0?I00%0L000:r0BT00^0
0       00z-00S?ub6-04wP0MT0000g00-0h002e?0005ij{0:+0mn!>0d
0i j '0CSG000000NTN1;00z-000y0B0)0000[*Δ0:t0000VQIMZ'0|v=000000
0       00,^N0Z0|00S?ub6-04wP0MT0000g00-0h002e?0005ij{0:+0mn!>0d
C0%0m90*0g000U0400*0Aw;0;0c0<o0p J00E;&0"00R0%0Z00000*PnQgsu0s20\]+0C4J00002000w000)00A0
00root@kali:~# gpg --decrypt test.txt
gpg: encrypted with 2048-bit RSA key, ID BC1CCE68DEDB589E, created 2019-10-24
"hunterdonald (COMP5370) <hxd0011@auburn.edu>
This is a test file to be encrypted and decrypted.
root@kali:~# cat test.txt
0
00h00x001a"qn000(000|0500+L0000>0cc000 $[000H&0w000:0]0R00FB0R0H[0_r00i0?I00%0L000:r0BT00^0
0       00z-00S?ub6-04wP0MT0000g00-0h002e?0005ij{0:+0mn!>0d
0i j '0CSG000000NTN1;00z-000y0B0)0000[*Δ0:t0000VQIMZ'0|v=000000
0       00,^N0Z0|00S?ub6-04wP0MT0000g00-0h002e?0005ij{0:+0mn!>0d
C0%0m90*0g000U0400*0Aw;0;0c0<o0p J00E;&0"00R0%0Z00000*PnQgsu0s20\]+0C4J00002000w000)00A0
00root@kali:~# gpg --decrypt test.txt
gpg: encrypted with 2048-bit RSA key, ID BC1CCE68DEDB589E, created 2019-10-24
"hunterdonald (COMP5370) <hxd0011@auburn.edu>
This is a test file to be encrypted and decrypted.
root@kali:~#
```

hunter@hunter-VirtualBox:~

```
File Edit View Search Terminal Help
```

```
hunter@hunter-VirtualBox:~$ wget http://192.168.1.6/hunterdonald.gpg
--2019-10-23 20:07:56-- http://192.168.1.6/hunterdonald.gpg
Connecting to 192.168.1.6:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1252 (1.2K)
Saving to: 'hunterdonald.gpg'

hunterdonald.gpg          100%[=====] 1.22K --.-KB/s   in 0s

2019-10-23 20:07:56 (199 MB/s) - 'hunterdonald.gpg' saved [1252/1252]

hunter@hunter-VirtualBox:~$ ls -ll
total 52
drwxr-xr-x 3 hunter hunter 4096 Sep  4 20:50 data
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Desktop
drwxr-xr-x 4 hunter hunter 4096 Sep  5 21:41 Documents
drwxr-xr-x 2 hunter hunter 4096 Sep  5 19:36 Downloads
-rw-r--r-- 1 hunter hunter 8980 Sep  1 23:44 examples.desktop
-rw-r--r-- 1 hunter hunter 1252 Oct 23 20:07 hunterdonald.gpg
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Music
drwxr-xr-x 2 hunter hunter 4096 Sep 16 15:19 Pictures
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Public
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Templates
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Videos
hunter@hunter-VirtualBox:~$ gpg --import hunterdonald.gpg
gpg: key A92AFA72100C2282: public key "hunterdonald (COMP5370) <hxd0011@auburn.edu>" imported
gpg: Total number processed: 1
gpg:           imported: 1
hunter@hunter-VirtualBox:~$ cat > test.txt
This is a test file to be encrypted and decrypted.
^C
hunter@hunter-VirtualBox:~$ gpg --encrypt --recipient hunterdonald hunterdonald.gpg test.txt
usage: gpg [options] --encrypt [filename]
hunter@hunter-VirtualBox:~$ gpg --encrypt --recipient hunterdonald test.txt
gpg: BC1CCE68DEDB589E: There is no assurance this key belongs to the named user
sub rsa2048/BC1CCE68DEDB589E 2019-10-24 hunterdonald (COMP5370) <hxd0011@auburn.edu>
  Primary key fingerprint: 87D6 D738 5367 E141 4E49  96F2 A92A FA72 100C 2282
  Subkey fingerprint: 6734 B159 A23F B159 CC59  A7C8 BC1C CE68 DEDB 589E

It is NOT certain that the key belongs to the person named
in the user ID. If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
hunter@hunter-VirtualBox:~$ ls -ll
total 52
```

Hunter Donald

hzd0011

COMP 5370

Homework 3

```
File Edit View Search Terminal Help
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Videos
hunter@hunter-VirtualBox:~$ gpg --import hunterdonald.gpg
gpg: key A92FA72100C2282: public key "hunterdonald (COMP5370) <hzd0011@auburn.edu>" imported
gpg: Total number processed: 1
gpg:          imported: 1
hunter@hunter-VirtualBox:~$ cat > test.txt
This is a test file to be encrypted and decrypted.
^C
hunter@hunter-VirtualBox:~$ gpg --encrypt --recipient hunterdonald hunterdonald.gpg test.txt
usage: gpg [options] --encrypt [filename]
hunter@hunter-VirtualBox:~$ gpg -e --recipient hunterdonald test.txt
gpg: BC1CE68DEDB589E: There is no assurance this key belongs to the named user
sub rsa2048/BC1CE68DEDB589E 2019-10-24 hunterdonald (COMP5370) <hzd0011@auburn.edu>
    Primary key fingerprint: 8706 D738 5367 E141 4E49  96F2 A92A FA72 100C 2282
    Subkey fingerprint: 6734 B159 A23F B159 CC59  A7C8 BC1C CE68 DEDB 589E

It is NOT certain that the key belongs to the person named
in the user ID. If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
hunter@hunter-VirtualBox:~$ ls -ll
total 60
drwxr-xr-x 3 hunter hunter 4096 Sep  4 20:50 data
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Desktop
drwxr-xr-x 4 hunter hunter 4096 Sep  5 21:41 Documents
drwxr-xr-x 2 hunter hunter 4096 Sep  5 19:36 Downloads
-rw-r--r--  1 hunter hunter  9890 Sep  1 23:44 examples.desktop
-rw-r--r--  1 hunter hunter 1252 Oct 23 20:07 hunterdonald.gpg
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Music
drwxr-xr-x 2 hunter hunter 4096 Sep 16 15:19 Pictures
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Public
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Templates
-rw-r--r--  1 hunter hunter   51 Oct 23 20:09 test.txt
-rw-r--r--  1 hunter hunter   382 Oct 23 20:11 test.txt.gpg
drwxr-xr-x 2 hunter hunter 4096 Sep  1 23:49 Videos
hunter@hunter-VirtualBox:~$ nc 192.168.1.6 < test.txt.gpg
usage: nc [-46CddFhklNnrStuvzz] [-I length] [-i interval] [-M ttl]
          [-m minttl] [-o length] [-P proxy_username] [-p source_port]
          [-q seconds] [-s source] [-t keyword] [-v rtable] [-W recvlimit] [-w timeout]
          [-x proxy_protocol] [-x proxy_address[:port]]           [destination] [port]
hunter@hunter-VirtualBox:~$ nc 192.168.1.6 4444 < test.txt.gpg
hunter@hunter-VirtualBox:~$ nc 192.168.1.6 4444 < test.txt.gpg
^C
hunter@hunter-VirtualBox:~$
```

3)

Original Data: 100110111010011

?1?001?10111010011

P<sub>1</sub> = ?101111101 = 1

P<sub>2</sub> = ?101010111 = 0

P<sub>4</sub> = ?0011101 = 0

P<sub>8</sub> = ?1011101 = 1

Encoded data: 1010001110111010011

Hunter Donald  
hzd0011  
COMP 5370  
Homework 3

4)

- a. If the key is known, take the location in the alphabet of the first letter of the ciphertext and subtract the location of the first letter of the key from that. If the result is negative, add the length of the alphabet to the result. The result of the subtraction will be the location in the alphabet of the first deciphered letter. Repeat this for the second letter of the ciphertext and the second letter of the key, then the third, and so on until the entire key has been used. Once the entire key has been used, go back to the first letter of the key and keep going until all of the ciphertext is deciphered.
- b.
  - i. For a known plaintext attack, the attacker has the original plaintext and the encrypted ciphertext. To determine the key, take the location of the first letter of the ciphertext and subtract the location of the first letter of the plaintext to get the location of the first letter of the key. If this value is negative, add the length of the alphabet to get the result. Repeat this for the second letter of the cipher text and plaintext, and so on until the entire key has been revealed. Once the key is revealed, the attacker is able to decrypt anything encrypted with that key using the same method described in problem 3 a.
  - ii. For a chosen ciphertext attack, the attacker can send the victim some ciphertext, and the victim will send back the plaintext. Now that the attacker has the ciphertext and its corresponding decrypted plaintext, the attacker can use the same method for a plaintext attack to determine the key.
  - iii. For a chosen plaintext attack, the attacker sends the victim some plaintext and the victim will send back the corresponding ciphertext. With this attack, the attacker is able to strategically choose the plaintext so that the key will be revealed more easily.
- c.
  - i. The attack I used to break this cipher was a simple brute force attack where I tried every possible key.
  - ii. To determine the key, since the ciphertext has a fair amount of two letter words, I declared an array with some of the most common two letter words. I then used that array to count the number of unique common words in the plaintext from each key and chose the key which yielded the highest number of unique common words in its respective plaintext.
  - iii. Key: AUB  
Deciphered text: TO BE OR NOT TO BE THAT IS THE QUESTION