Vidyalankar Institute of Technology

Wadala(E), Mumbai 400037

CERTIFICATE

This is to certify that this Lab Work (ISA) in the subject of

Cryptography and System Security

of semester **VI** of **Computer Engineering** course (**Academic Year 2023-24**)

Submitted by

Mr .Deep Salunkhe Roll No 21102A0014

is accepted by the Department

Amit K. Nerurkar

Professor, In Charge

DEPARTMENT OF COMPUTER ENGINEERING

Semester	T.E. Semester VI-CMPN (DIVA B1)
Subject	Cryptography and System Security
Professor In-charge	Prof. Amit K. Nerurkar
Laboratory	M 312B

Index

Sr. No.	Experiment Title	Date
1	Design and Implementation of Ceaser Cipher Technique	17/1/24
2	Design and Implementation of Monoalphabetic Cipher	24/1/24
3	Design and Implementation of Digital Signature	31/1/24
4	Design and Implementation of RSA algorithm for generating public and private key	14/2/24
5	Design and Implementation of DES (Symmetric Key Encryption)	21/2/24
6	Special Topic Seminar	6/3/24
7	Design and Implementation of Diffie Hellman Key Establishment	13/3/24
8	Design and Implementation of HMAC	20/3/24
9	Design and Implementation of DOS using Hping 3	27/3/24
10	Simulation of SQL Injection (PBLE-1)	3/4/24
11	One of the most famous intrusion detection system and has been known for its flexibility with different environments. You can even integrate it with Kibana and elastic cloud. PBLE-2	10/4/24
12	Mini Project	24/4/24

Amit K. Nerurkar

Subject In-charge

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

DEPARTMENT OF COMPUTER ENGINEERING
CSS Lab

Semester	T.E. Semester V I– Computer Engineering
Subject	Cryptography and System Security
Subject Professor In-charge	Prof. Amit K. Nerurkar
Assisting Teachers	Prof. Amit K. Nerurkar
Laboratory	312A

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

Design and Implementation of Ceaser Cipher Technique

Explanation:

A private-key encryption scheme consists of a set of all possible messages, called the message space M, and three algorithms, namely,

(a) Gen

(b) Enc

(c) Dec

The algorithm for key generation Gen is used to choose a key k at random from the set of all possible secert keys, denoted by the key space K.

The algorithm for encryption Enc takes as inputs the message m and the secret key k and outputs the ciphertext c.

The algorithm for decryption Dec inputs the ciphertext c and the key k and outputs the message m.

About the experiment:

Apparently, the system is easily broken if the total number of distinct secret keys is small, that is the key space K is small.

In this experiment, we work with a well-known historical encryption scheme, namely the shift cipher, that has a very small key space.

Your task is to break the shift cipher. Specifically, given (only) the ciphertext in some instance of a shift cipher, you need to find the plaintext and the secret key.

DEPARTMENT OF COMPUTER ENGINEERING
CSS Lab

Alphabets	Positions
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8
J	9
K	10
L	11
M	12
N	13
O	14
P	15
Q	16
R	17
S	18
T	19
U	20
V	21
W	22
X	23
Y	24
Z	25

Plain text	P	r	e	m	a	n	s	h	u	u
Position	15	17	4	12	0	13	18	7	20	20
Key	19	19	19	19	19	19	19	19	19	19
P+K	34	36	23	31	19	32	37	26	39	39
(P+K)%26	8	10	23	5	19	6	11	0	13	13
Cipher text	I	K	X	F	T	G	L	A	N	N

Cipher text	I	K	X	F	T	G	L	A	N	N
Positions	8	10	23	5	19	6	11	0	13	13
Key	19	19	19	19	19	19	19	19	19	19
P-k	-11	-9	4	-14	0	-13	-8	-19	-6	-6
26+(P-k)	15	17		12		13	18	7	20	20
Plain Text	P	r	e	m	a	n	s	h	u	u

Note:26 +(p-k) is applicable only when p-k is -ve

Alphabets	Positions
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8
J	9
K	10
L	11
M	12
N	13
O	14
P	15
Q	16
R	17
S	18
T	19
U	20
V	21
W	22
X	23
Y	24
Z	25

Plain text	P	r	e	m	a	n	s	h	u	u
Position	15	17	4	12	0	13	18	7	20	20
Key	19	19	19	19	19	19	19	19	19	19
P*K	285	323	76	228	0	247	342	133	380	380
(P*K)%26	25	11	24	20	0	13	4	3	16	16
Cipher text	Z	L	Y	U	A	N	E	D	Q	Q

k inverse=11

Cipher text	Z	L	Y	U	A	N	E	D	Q	Q
Position	25	11	24	20	0	13	4	3	16	16
Key	19	19	19	19	19	19	19	19	19	19
(P-k(inverse))%26	15	17	4	12	0	13	18	7	20	20
Plain text	P	r	e	m	a	n	s	h	u	u

Simulation:

PART I

Ciphertext to be decrypted:

```
WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ
```

Next Ciphertext

PART II

Do your rough work here:

```
wkh txlfn eurzq ira mxpsv ryhu wkh odcb grj = 0
vjg swkem dtqyp hqz lworu qxgt vjg ncba fqi = 1
uif rvjdl cspxo gpy kvnqt pwfs uif mbaz eph = 2
the quick brown fox jumps over the lazy dog = 3
```

PART III

Plaintext:

```
the quick brown fox jumps over the lazy dog
```

shift: 3

v Encrypt v | ^ Decrypt ^

Ciphertext

```
WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ
```

PART IV

Enter your solution Plaintext and shift key here:

the quick brown fox jumps over the lazy dog

Key 3

Check my answer!

CORRECT!!

Conclusion:

In conclusion, the experiment successfully broke the shift cipher, showcasing its vulnerability due to its small key space. By systematically trying all possible keys and analyzing letter frequencies, the plaintext message was deciphered. This highlights the importance of key space size in encryption security and underscores the need for robust cryptographic techniques.

Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title: Design and Implementation of Monoalphabetic Cipher

Explanation:

A monoalphabetic substitution cipher is a simple form of encryption where each letter in the plaintext is replaced by a corresponding letter in the ciphertext according to a fixed substitution scheme. In other words, each letter is consistently replaced by another letter throughout the message.

Breaking a monoalphabetic substitution cipher involves identifying the substitution key or pattern used to encrypt the message so that it can be decrypted. Here's a general approach to breaking such a cipher:

1. Frequency Analysis: Since the encryption scheme replaces each letter with another letter, the frequency of letters in the ciphertext should roughly correspond to the frequency of letters in the plaintext language. For instance, in English, 'E' is the most common letter, followed by 'T', 'A', and so on. By analyzing the frequency of letters in the ciphertext, one can make educated guesses about the substitutions.

2. Single-Letter Words: In English, the most common single-letter word is 'I'. If a single-letter word appears frequently in the ciphertext, it's likely to correspond to 'I' in the plaintext. Similarly, if a three-letter word appears, it's often 'THE'. Using such patterns can help deduce some letters.

3. Pattern Recognition: Look for recurring patterns of letters in the ciphertext, which may correspond to common words, prefixes, or suffixes in the plaintext. For example, 'TH', 'ING', 'TION', etc., are common patterns in English.

4. Contextual Analysis: If part of the message is known or can be guessed, it can provide clues about the substitutions. For instance, if the encrypted message is likely to contain certain common words or phrases (like "Dear," "Sincerely," etc.), identifying these can help determine their corresponding ciphertext letters.

5. Trial and Error: Use a combination of the above techniques to make educated guesses about the substitutions. You can start with the most frequently occurring letters and work your way down. As more letters are decrypted, it becomes easier to decipher the rest of the message.

6. Iterative Refinement: As more letters are decrypted, refine the substitutions and reanalyze the ciphertext for additional patterns and clues. This iterative process continues until the entire message is decrypted.

7. Manual or Automated Methods: Breaking a monoalphabetic substitution cipher can be done manually or with the help of computer algorithms. Automated methods can significantly speed up the process, especially for longer messages.

Simulation:

PART I

Decrypt the following cipher text. A tool to simulate the Mono-Alphabetic Subsitution cipher is provided beneath for your assistance.

Here is the table of frequencies of English alphabets for your reference:

a	b	c	d	e	f	g	h	i	j	k	l	m
8.167	1.49	2.782	4.253	12.702	2.228	2.015	6.094	6.966	0.153	0.772	4.025	2.406
n	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
6.749	7.507	1.929	0.095	5.987	6.327	9.056	2.758	0.978	2.360	0.150	1.974	0.074

```
dkxyvrh 1 - qegt vkr hxccwv keur: xuwdr wn cehrq nwvvwtp et vkr
hwsrhcxto gwvk krh nwnvrh, gkrt nkr tevwdrn x vxuowtp, duevkrq gkwvr
hxccwv gwvk x yedorv gxvdk hit yxnv. nkr leuuegn wv qegt x hxccwv keur
gkrt niqqrtub nkr lxuun x uetp gxb ve x dihwein kxuu gwvk fxtb uedorq
qeehn el xuu nwmrn. nkr lwtqn x nfxuu orb ve x qeeh vee nfxuu leh krh
ve lwv, civ vkheipk gkwdk nkr nrrn xt xvvhxdvwsr pxhqrt. nkr vkrt
```

Next Ciphertext

Calculate Frequencies in ciphertext

Ciphertext Frequencies:

a	b	c	d	e	f	g	h	i	j	k	l	m
0.000	1.037	2.282	3.942	8.091	1.452	3.112	5.602	2.075	0.000	8.506	1.452	0.415
n	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
7.469	1.867	1.452	3.32	11.618	0.622	4.979	5.602	9.959	6.639	7.884	0.622	0.000

PART II

Note that the *cipher text is in lower case* and when you replace any character, the final character of replacement, i.e., *plaintext is changed to upper case* automatically in the following scratchpad.

Scratchpad:

```
CHAPTER 1 - DOWN THE RABBIT HOLE: ALICE IS BORED SITTING ON THE
RIVERBANK WITH HER SISTER, WHEN SHE NOTICES A TALKING, CLOTHED WHITE
RABBIT WITH A POCKET WATCH RUN PAST. SHE FOLLOWS IT DOWN A RABBIT HOLE
WHEN SUDDENLY SHE FALLS A LONG WAY TO A CURIOUS HALL WITH MANY LOCKED
DOORS OF ALL SIZES. SHE FINDS A SMALL KEY TO A DOOR TOO SMALL FOR HER TO
FIT, BUT THROUGH WHICH SHE SEES AN ATTRACTIVE GARDEN. SHE THEN DISCOVERS
A BOTTLE LABELLED 'DRINK ME', THE CONTENTS OF WHICH CAUSE HER TO SHRINK
TOO SMALL TO REACH THE KEY. A CAKE WITH 'EAT ME' ON IT CAUSES HER TO
GROW TO SUCH A TREMENDOUS SIZE HER HEAD HITS THE CEILING.
```

Modify the text above (in scratchpad):

This is case *insensitive* function and replaces only cipher text (lower case) by plain text (upper case):

Replace cipher character [m] by plaintext character [z] Modify

Use the following function to undo any unwanted exchange by giving an uppercase character and a lower case. This is a case sensitive function:

Replace character [ ] by character [ ] Replace these exact characters

Your replacement history:

You replaced x by A You replaced c by B You replaced d by C You replaced q by D You replaced r by E You replaced l by F You replaced p by G You replaced k by H You replaced w by I You replaced z by J You replaced o by K You replaced u by L You replaced f by M You replaced t by N You replaced e by O You replaced y by P You replaced a by Q You replaced h by R You replaced n by S You replaced v by T You replaced i by U You replaced s by V You replaced g by W You replaced j by X You replaced b by Y You replaced m by Z

PART III

Enter your solution plaintext here:

```
TO FIT, BUT THROUGH WHICH SHE SEES AN ATTRACTIVE GARDEN. SHE THEN
DISCOVERS A BOTTLE LABELLED 'DRINK ME', THE CONTENTS OF WHICH CAUSE
HER TO SHRINK TOO SMALL TO REACH THE KEY. A CAKE WITH 'EAT ME' ON IT
CAUSES HER TO GROW TO SUCH A TREMENDOUS SIZE HER HEAD HITS THE
CEILING.
```

Solution Key = xcdqrlpkwzoufteyahnvisgjbm

Check Answer!

```
CORRECT!!
```

PART IV

Plaintext

```
with 'eat me' on it causes her to grow to such a
tremendous size her head hits the ceiling.
```

key = xcdqrlpkwzoufteyahnvisgjbm [Generate Random Key]

[v Encrypt v] [^ Decrypt ^]

☐ Remove Punctuation

Ciphertext

```
dkxyvrh 1 - qegt vkr hxccwv keur: xuwdr wn cehrq
nwvvwtp et vkr hwsrhcxto gwvk krh nwnvrh, gkrt nkr
```

Conclusion:

In conclusion, breaking a monoalphabetic substitution cipher involves analyzing the frequency, patterns, and context of the ciphertext to deduce the substitutions used in the encryption. By employing techniques such as frequency analysis, identifying single-letter words, recognizing patterns, considering contextual clues, and employing trial and error, it's possible to decrypt the message. This process may require manual effort or the use of automated methods, depending on the complexity of the cipher and the available resources. Ultimately, with patience, persistence, and careful analysis, the encryption key or pattern can be determined, allowing for the decryption of the message.

Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

Design and Implementation of Digital Signature

Explanation:

1. Definition: A digital signature is a cryptographic technique used to verify the authenticity and integrity of digital messages or documents. It provides a way for the sender of a message to prove their identity and ensure that the message has not been altered in transit.
2. Components of Digital Signatures:
    - Signing Algorithm: A mathematical algorithm used by the sender to generate the digital signature from the message.
    - Verification Algorithm: A complementary algorithm used by the recipient to verify the authenticity and integrity of the message and signature.
    - Key Pair: Digital signatures are typically based on asymmetric cryptography, where the sender possesses a private key for signing and the recipient uses a corresponding public key for verification.
    - Hash Function: A cryptographic hash function is often used to generate a fixed-size hash value from the message before signing. This hash value ensures that the signature is based on the content of the message and cannot be used to reconstruct the original message.
3. Process of Creating a Digital Signature:
    - The sender computes a hash value of the message using a secure hash function.
    - The sender encrypts the hash value with their private key, generating the digital signature.
    - The sender sends both the original message and the digital signature to the recipient.
4. Process of Verifying a Digital Signature:
    - The recipient computes a hash value of the received message using the same secure hash function used by the sender.

- The recipient decrypts the digital signature using the sender's public key, obtaining the original hash value.
- The recipient compares the computed hash value with the decrypted hash value. If they match, the signature is valid, and the message is authentic and unaltered.

5. Properties of Digital Signatures:
    - Authentication: Digital signatures authenticate the identity of the sender, ensuring that the message originates from a known and trusted source.
    - Integrity: Digital signatures verify that the message has not been altered or tampered with during transmission.
    - Non-repudiation: Digital signatures provide non-repudiation, meaning that the sender cannot deny sending the message once it has been digitally signed and verified.
    - Unforgeability: A valid digital signature cannot be forged by an unauthorized party, as it requires possession of the sender's private key.
6. Applications:
    - Digital signatures are widely used in electronic transactions, digital contracts, secure email communication, software distribution, and other scenarios where authentication and integrity are critical.
    - They are essential for ensuring trust and security in digital environments, especially in situations where physical signatures are impractical or impossible.

Result:

RSA public key

Public exponent (hex, F4=0x10001):

10001

Modulus (hex):

```
a5261939975948bb7a58dffe5ff54e65f0498f9175f5a09288810b8975871e99
af3b5dd94057b0fc07535f5f97444504fa35169d461d0d30cf0192e307727c06
5168c788771c561a9400fb49175e9e6aa4e23fe11af69e9412dd23b0cb6684c4
c2429bce139e848ab26d0829073351f4acd36074eafd036a5eb83359d2a698d3
```

1024 bit | 1024 bit (e=3) | 512 bit | 512 bit (e=3)

Digitally sign the plaintext with Hashed RSA.

Plaintext (string):

test SHA-1

Hash output(hex):

a94a8fe5ccb19ba61c4c0873d391e987982fbbd3

Input to RSA(hex):

a94a8fe5ccb19ba61c4c0873d391e987982fbbd3 Apply RSA

Digital Signature(hex):

```
2576257c9fa21a134f6e82fa9ffced2ff85b629146e77e6d601eb8b97420798d
814e697f7e081a6a3949b31ecdeb008566d420c2e84926b1531bf695c78e5421
1821948b3f3fb9a239ee642298ce4529b4918270d2f18d47de35ec55678ad5c6
f66616727869d79b87102e1126d991be9f23f8da2cbb27a0197ed50d72fdb3c5
```

Digital Signature(base64):

```
JXYlfJ+iGhNPboL6n/ztL/hbYpFG535tYB64uXQgeY2BTml/fggaajlJsx7N6wCF
ZtQgwuhJJrFTG/aVx45UIRghlIs/P7miOe5kIpjORSm0kYJw0vGNR9417FVnitXG
9mYWcnhp15uHEC4RJtmRvp8j+NosuyegGX7VDXL9s8U=
```

Status:

Time: 2ms

Conclusion:

By understanding these theoretical aspects of digital signatures, students can gain insights into their importance, functionality, and applications in cryptography and system security. Lab exercises can involve implementing digital signature algorithms, experimenting with different

parameters and key sizes, and exploring real-world use cases to reinforce learning and understanding.

Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title : **Design and Implementation of RSA algorithm for generating public and private key**

Title:

Design and Implementation of RSA algorithm for generating public and private key

---

Explanation:

1. Definition: A public-key cryptosystem, also known as asymmetric cryptography, is a cryptographic technique that uses two keys - a public key and a private key - to perform encryption and decryption operations.
2. Key Generation:
    - Public Key: The public key is made freely available to anyone and is used for encryption by other parties.
    - Private Key: The private key is kept secret by its owner and is used for decryption. It must never be shared with others.
3. Encryption and Decryption:
    - Encryption: A message or data is encrypted using the recipient's public key. Only the corresponding private key can decrypt the ciphertext.
    - Decryption: The encrypted message is decrypted using the recipient's private key, ensuring that only the intended recipient can access the original plaintext.
4. Security Properties:
    - Confidentiality: Public-key cryptosystems provide confidentiality by ensuring that only the intended recipient, who possesses the corresponding private key, can decrypt and access the original message.
    - Authentication: Public-key cryptosystems enable authentication by allowing users to digitally sign messages using their private keys, which can be verified by anyone using the corresponding public key.
    - Integrity: Digital signatures generated using public-key cryptography also ensure data integrity, as any tampering with the message will result in an invalid signature.

- Non-repudiation: Public-key cryptosystems provide non-repudiation, meaning that a user cannot deny sending a message or creating a digital signature once it has been verified using their public key.

5. Applications:

   - Secure Communication: Public-key cryptosystems are used to establish secure communication channels over insecure networks, such as the internet. Examples include HTTPS for secure web browsing and S/MIME for secure email communication.
   - Digital Signatures: Public-key cryptography is used to generate and verify digital signatures, ensuring the authenticity and integrity of electronic documents, transactions, and messages.
   - Key Exchange: Public-key cryptosystems facilitate secure key exchange protocols, such as Diffie-Hellman key exchange, which allows parties to establish a shared secret key over an insecure channel.

6. Security Considerations:

   - Key Management: Proper key management practices, including key generation, distribution, storage, and revocation, are crucial for the security of public-key cryptosystems.
   - Key Sizes: The security of public-key cryptosystems depends on the size of the keys used. Larger key sizes provide stronger security but may impact performance.
   - Algorithm Selection: Choosing secure and widely-accepted cryptographic algorithms, such as RSA, ECC, or ElGamal, is essential to ensure the security and interoperability of public-key cryptosystems.

Result:

RSA private key

1024 bit | 1024 bit (e=3) | 512 bit | 512 bit (e=3) | Generate | bits = 512

Modulus (hex):

```
a5261939975948bb7a58dffe5ff54e65f0498f9175f5a09288810b8975871e99
af3b5dd94057b0fc07535f5f97444504fa35169d461d0d30cf0192e307727c06
5168c788771c561a9400fb49175e9e6aa4e23fe11af69e9412dd23b0cb6684c4
c2429bce139e848ab26d0829073351f4acd36074eafd036a5eb83359d2a698d3
```

Public exponent (hex, F4=0x10001):

```
10001
```

Private exponent (hex):

```
8e9912f6d3645894e8d38cb58c0db81ff516cf4c7e5a14c7f1eddb1459d2cded
4d8d293fc97aee6aefb861859c8b6a3d1dfe710463e1f9ddc72048c09751971c
4a580aa51eb523357a3cc48d31cfad1d4a165066ed92d4748fb6571211da5cb1
4bc11b6e2df7c1a559e6d5ac1cd5c94703a22891464fba23d0d965086277a161
```

P (hex):

```
d090ce58a92c75233a6486cb0a9209bf3583b64f540c76f5294bb97d285eed33
aec220bde14b2417951178ac152ceab6da7090905b478195498b352048f15e7d
```

Q (hex):

```
cab575dc652bb66df15a0359609d51d1db184750c00c6698b90ef3465c996551
03edbf0d54c56aec0ce3c4d22592338092a126a0cc49f65a4a30d222b411e58f
```

D mod (P-1) (hex):

```
1a24bca8e273df2f0e47c199bbf678604e7df7215480c77c8db39f49b000ce2c
f7500038acfff5433b7d582a01f1826e6f4d42e1c57f5e1fef7b12aabc59fd25
```

D mod (Q-1) (hex):

```
3d06982efbbe47339e1f6d36b1216b8a741d410b0c662f54f7118b27b9a4ec9d
914337eb39841d8666f3034408cf94f5b62f11c402fc994fe15a05493150d9fd
```

1/Q mod P (hex):

```
3a3e731acd8960b7ff9eb81a7ff93bd1cfa74cbd56987db58b4594fb09c09084
db1734c8143f98b602b981aaa9243ca28deb69b5b280ee8dcee0fd2625e53250
```

Title : Design and Implementation of RSA algorithm for generating public and private key

Roll No: 21102A0014

Plaintext (string):

deep

encrypt

Ciphertext (hex):

```
542132f674bb35f26f56282e21331596837b8cb41c2edacd2439f682cd77011f
17b2eebb8eb10baa4cbad7def3563fe6dab3d0880fdb1f131762d0b96cee9819
c2fec8984a56ceb3f7e1e436df6aa3001621e0b2f507e3c948cea1c91691841b
368e37b408a581b7111379dc6e1f2c0c3afc6a7908688316cc00f32e3139bd3f
```

decrypt

Decrypted Plaintext (string):

deep

Status:

Decryption Time: 6ms

Conclusion:

By understanding these theoretical aspects of public-key cryptosystems, students can gain insights into their principles, functionalities, and applications in cryptography and system security. Lab exercises can involve implementing encryption, decryption, digital signature generation and verification, key exchange protocols, and exploring real-world use cases to reinforce learning and understanding.

Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title: Design and Implementation of DES (Symmetric Key Encryption)

Title:

Design and Implementation of DES (Symmetric Key Encryption)

Explanation:

DES (Data Encryption Standard) is a symmetric key encryption algorithm that was developed in the 1970s by IBM and eventually adopted by the U.S. government as a federal standard for encrypting sensitive but unclassified information. It has since been widely used in various applications, although its security is now considered inadequate against modern cryptographic attacks due to its relatively short key length of 56 bits.

Here's an overview of DES and its types:

1. DES (Data Encryption Standard) :

- The original DES algorithm operates on 64-bit blocks of plaintext using a 56-bit key. It goes through a series of 16 rounds of substitution and permutation (known as the Feistel cipher structure) to produce the ciphertext. Each round uses a different 48-bit subkey derived from the original 56-bit key.

- Despite its widespread use in the past, DES is now considered insecure against brute-force attacks due to its small key space. It is vulnerable to attacks that exploit its short key length, such as exhaustive key search.

2. 3DES (Triple DES) :

- To address the security weaknesses of DES, 3DES was introduced. It applies the DES algorithm three times sequentially, using two or three different keys. The three-key variant of 3DES provides significantly stronger security than DES, as it effectively uses a key length of 168 bits (three 56-bit keys).

- While 3DES improves security, it is slower and requires more computational resources compared to DES due to the increased number of rounds.

3. DES Variants and Modes :

- DESX : A variant of DES that involves XORing the plaintext with some key material before and after encryption. This is used to increase resistance against certain attacks.

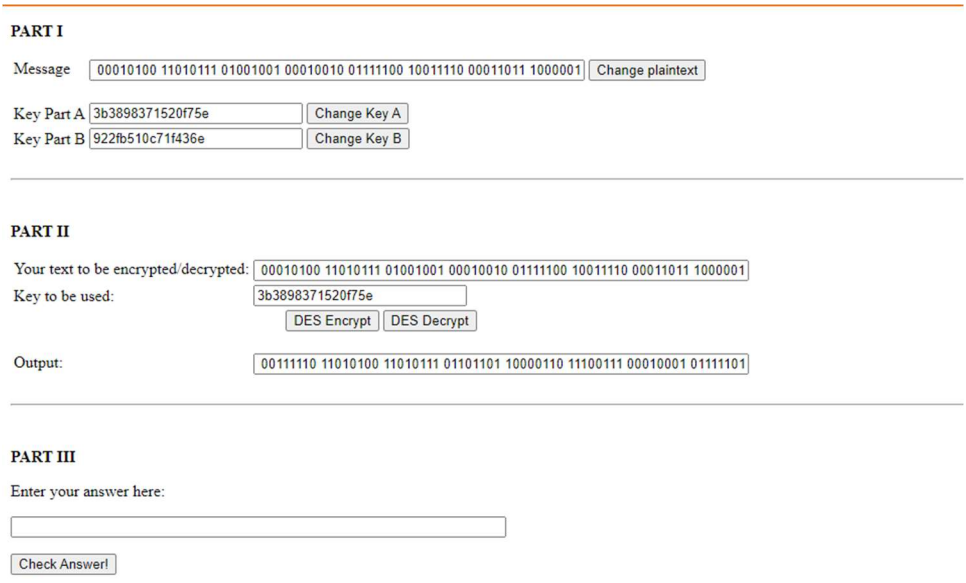
- Modes of Operation : DES can be used in different modes of operation, such as ECB (Electronic Codebook), CBC (Cipher Block Chaining), CFB (Cipher Feedback), OFB (Output Feedback), and CTR (Counter). These modes dictate how the encryption process is applied to plaintext blocks, and they have implications for security and performance in different scenarios.

4. DES Cryptanalysis:

- Over the years, various cryptanalytic techniques have been developed to exploit weaknesses in DES. Differential and linear cryptanalysis are among the most notable techniques used to analyze the security of DES and its variants.

- These attacks exploit patterns in plaintext-ciphertext pairs to recover the encryption key or reduce the effective key space, thereby making brute-force attacks more feasible.

Result:



Title: Design and Implementation of DES (Symmetric Key Encryption)

PART I

Message 00010100 11010111 01001001 00010010 01111100 10011110 00011011 1000001 Change plaintext

Key Part A 3b3898371520f75e Change Key A
Key Part B 922fb510c71f436e Change Key B

PART II

Your text to be encrypted/decrypted: 00111110 11010100 11010111 01101101 10000110 11100111 00010001 01111101
Key to be used: 922fb510c71f436e
DES Encrypt DES Decrypt

Output: 01001111 10110010 00100010 10101110 11101000 11001101 10010011 1001101

PART III

Enter your answer here:

Check Answer!

PART I

Message 00010100 11010111 01001001 00010010 01111100 10011110 00011011 1000001 Change plaintext

Key Part A 3b3898371520f75e Change Key A
Key Part B 922fb510c71f436e Change Key B

PART II

Your text to be encrypted/decrypted: 00111110 11010100 11010111 01101101 10000110 11100111 00010001 01111101
Key to be used: 3b3898371520f75e
DES Encrypt DES Decrypt

Output: 00010100 11010111 01001001 00010010 01111100 10011110 00011011 1000001

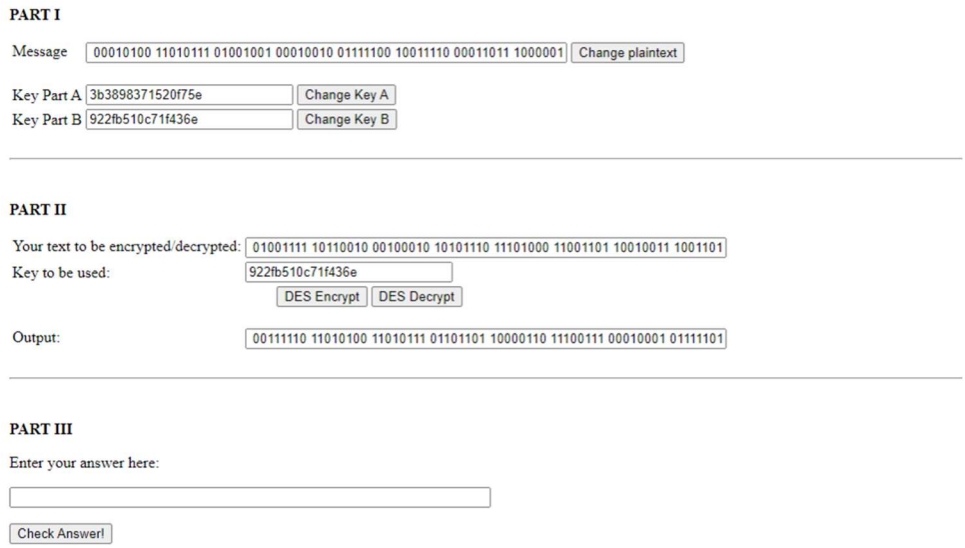
PART III

Enter your answer here:

Check Answer!

Title: Design and Implementation of DES (Symmetric Key Encryption)

Roll No: 21102A0014



Conclusion:

In conclusion, DES (Data Encryption Standard) is a foundational symmetric key encryption algorithm that has been widely used for decades. However, its security is now inadequate due to its small key size of 56 bits, which makes it vulnerable to brute-force attacks. To address these weaknesses, variants like Triple DES (3DES) were introduced, which apply the DES algorithm multiple times with different keys. Despite this, 3DES is slower and less efficient compared to modern encryption standards.

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

DEPARTMENT OF COMPUTER ENGINEERING
CSS Lab

Semester	T.E. Semester V I– Computer Engineering
Subject	Cryptography and System Security
Subject Professor In-charge	Prof. Amit K. Nerurkar
Assisting Teachers	Prof. Amit K. Nerurkar
Laboratory	Lab number 312 B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

IDS TECHNOLOGIES

Sahil Pokharkar (21102A0006)
Deep Salunkhee (21102A0014)
Omkar Patil (21102A0003)
Pranav Redij (21102A0005)

OBJECTIVES

Section 1: Introduction to IDS Technologies

- Overview of IDS technologies
- Typical components
- Network architectures in the context of IDS

Section 2: Security Capabilities of IDS

- Security Capabilities
  - Information Gathering
  - Logging Capabilities
  - Detection & Prevention Capabilities

SECTION 3: ADVANCED IDS TECHNOLOGIES

- Intrusion Prevention Systems (IPS)
- Network Protocol based IDS
- Hybrid based IDS

Section 4: Analysis Schemes for Intrusion

- Analysis Schemes for Instrusion
- Model Intrusion analysis

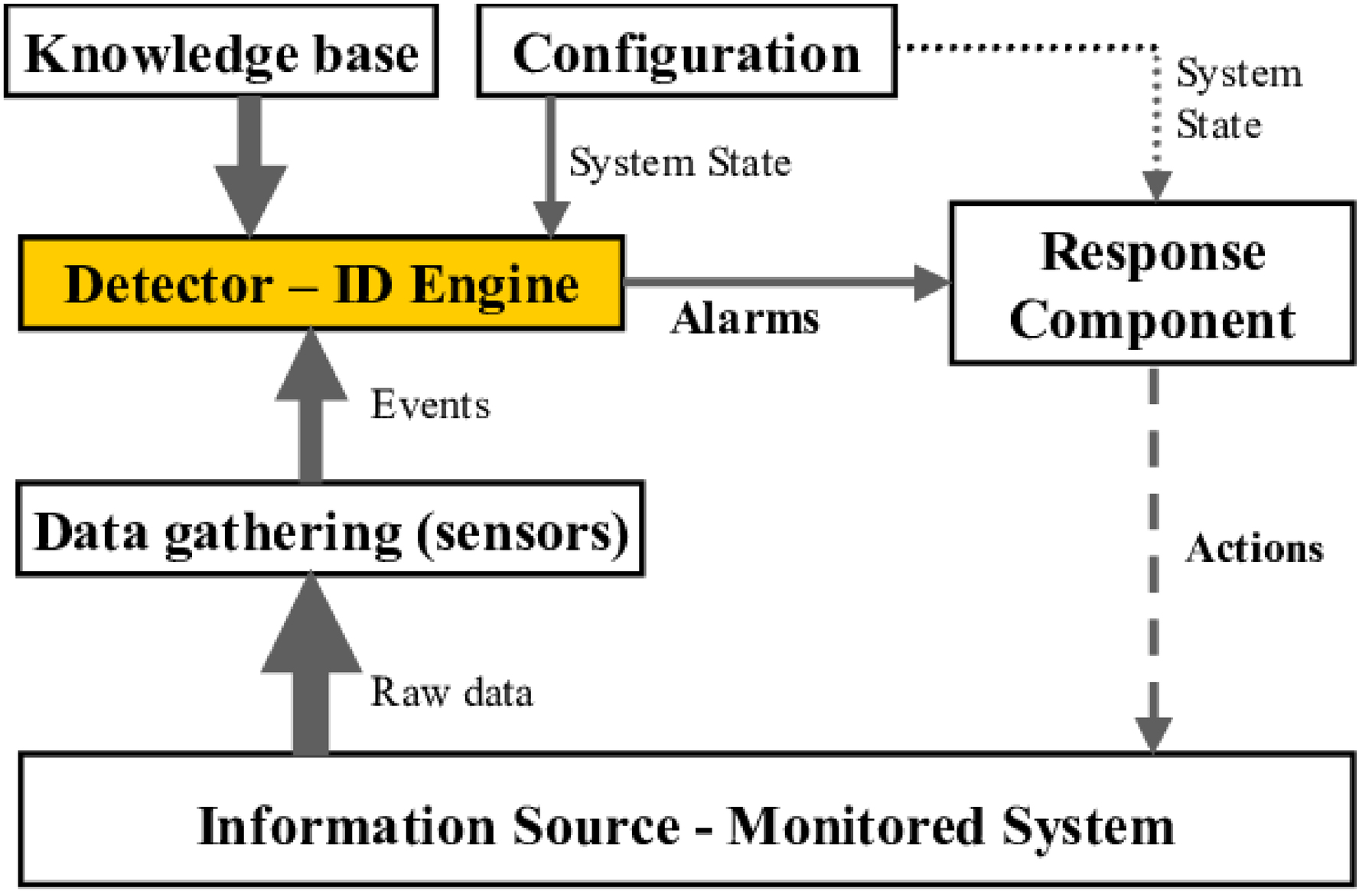
Section 5: Security Assessments

- Techniques for Intrusion Analysis
- Mapping Responses to Policy
- Vulnerability Analysis
- Credential Analysis

SECTION 1: INTRODUCTION TO IDS TECHNOLOGIES

1.1 OVERVIEW OF IDS TECHNOLOGIES

Intrusion Detection System (IDS) is a security technology that monitors network or system activities for malicious or suspicious behavior. It identifies potential security threats, such as unauthorized access attempts, malware infections, or abnormal network traffic patterns, by analyzing collected data against predefined rules or signatures. IDS helps organizations detect and respond to security incidents promptly, enhancing overall cybersecurity posture

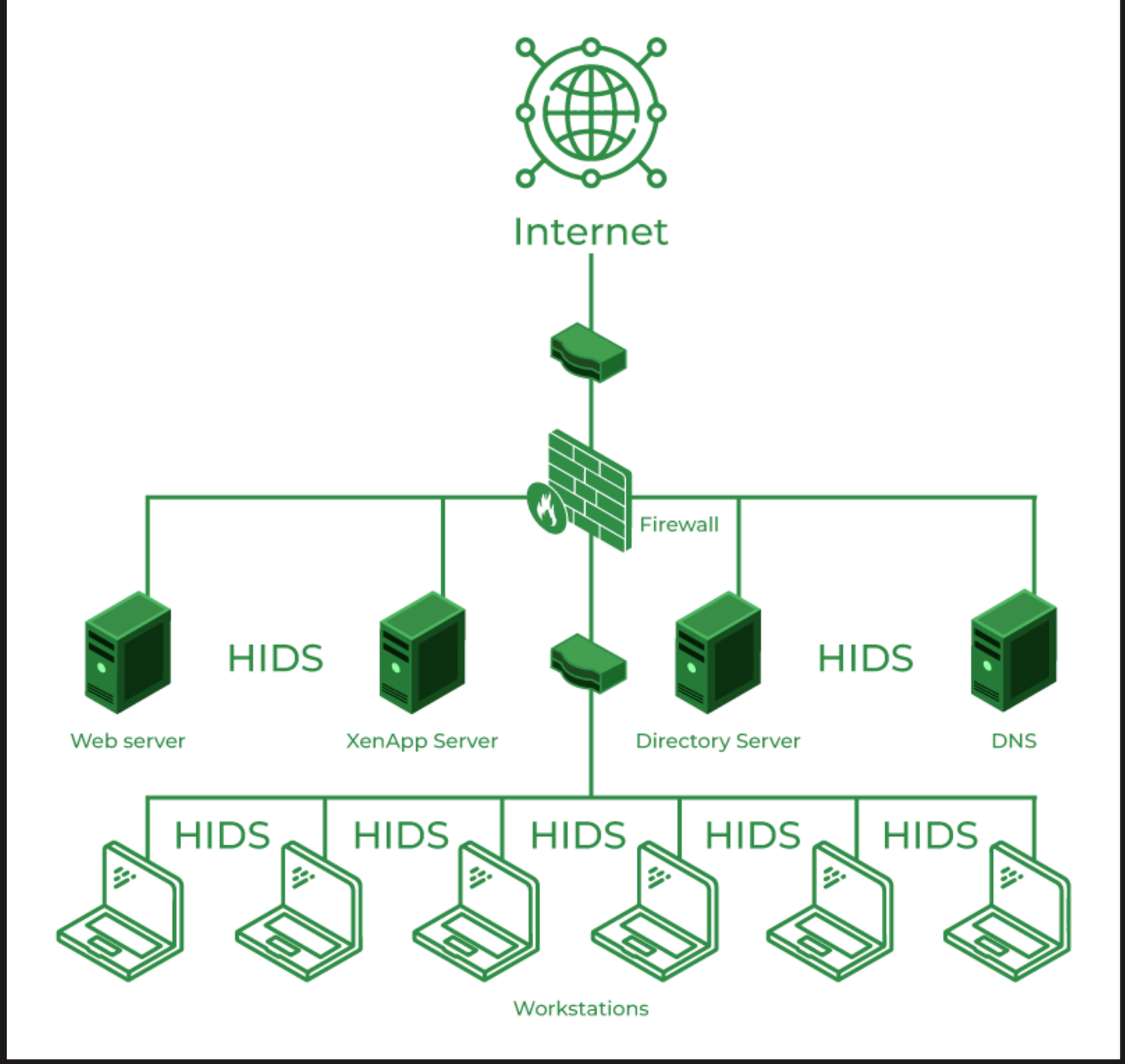

Knowledge base
Configuration
System State
System State
Detector – ID Engine
Alarms
Response Component
Events
Data gathering (sensors)
Actions
Raw data
Information Source - Monitored System

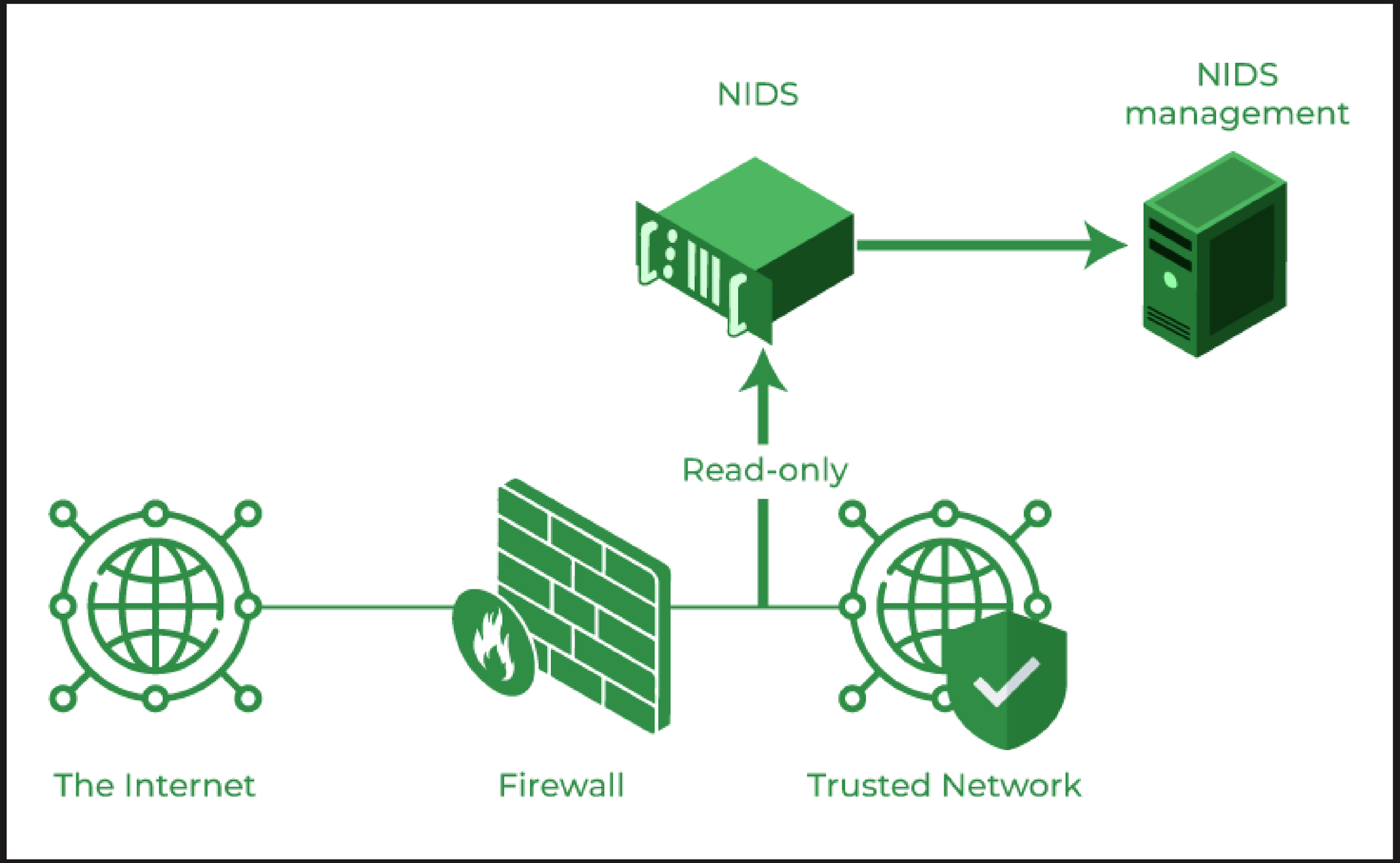
1.2 TYPICAL COMPONENTS

sensors (hardware or software)
analyzers
consoles

1.3 NETWORK ARCHITECTURES IN THE CONTEXT OF IDS

host-based
network-based
hybrid IDS setups


Internet
Firewall
HIDS
HIDS
Web server
XenApp Server
Directory Server
DNS
HIDS
HIDS
HIDS
HIDS
HIDS
Workstations


NIDS
NIDS management
Read-only
The Internet
Firewall
Trusted Network

SECTION 2: SECURITY CAPABILITIES OF IDS

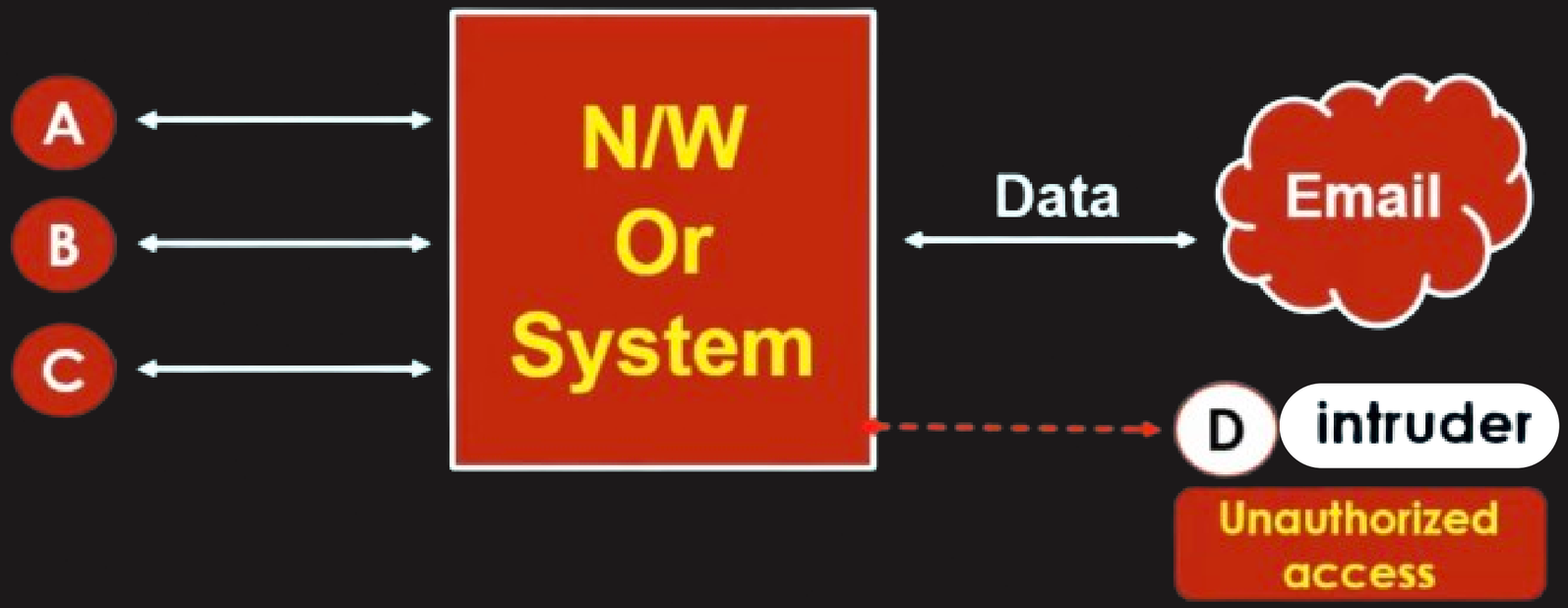
2.1 SECURITY CAPABILITIES

- Information Gathering
- Logging Capabilities
- Detection & Prevention Capabilities

SECTION 3: ADVANCED IDS TECHNOLOGIES

INTRUSION

intruder’s -> intrusion
(ghuskhor) (ghuskhori kar raha)



Real life and funny example to understand

difference

B/W

FIREWALL,IDS and IPS

Intruder’s

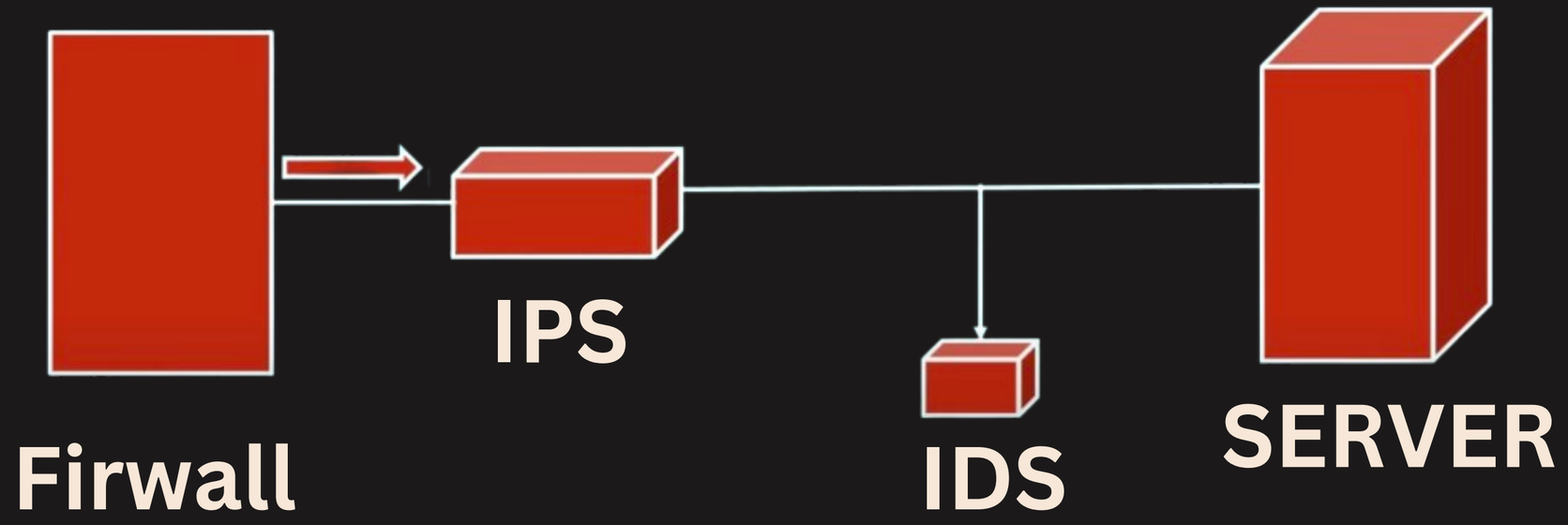

FIREWALL
they are INTRUDER'S
they are
AUTHORIZED USERS
अरे! ओ उंकल

IDS(intrusion detection system)

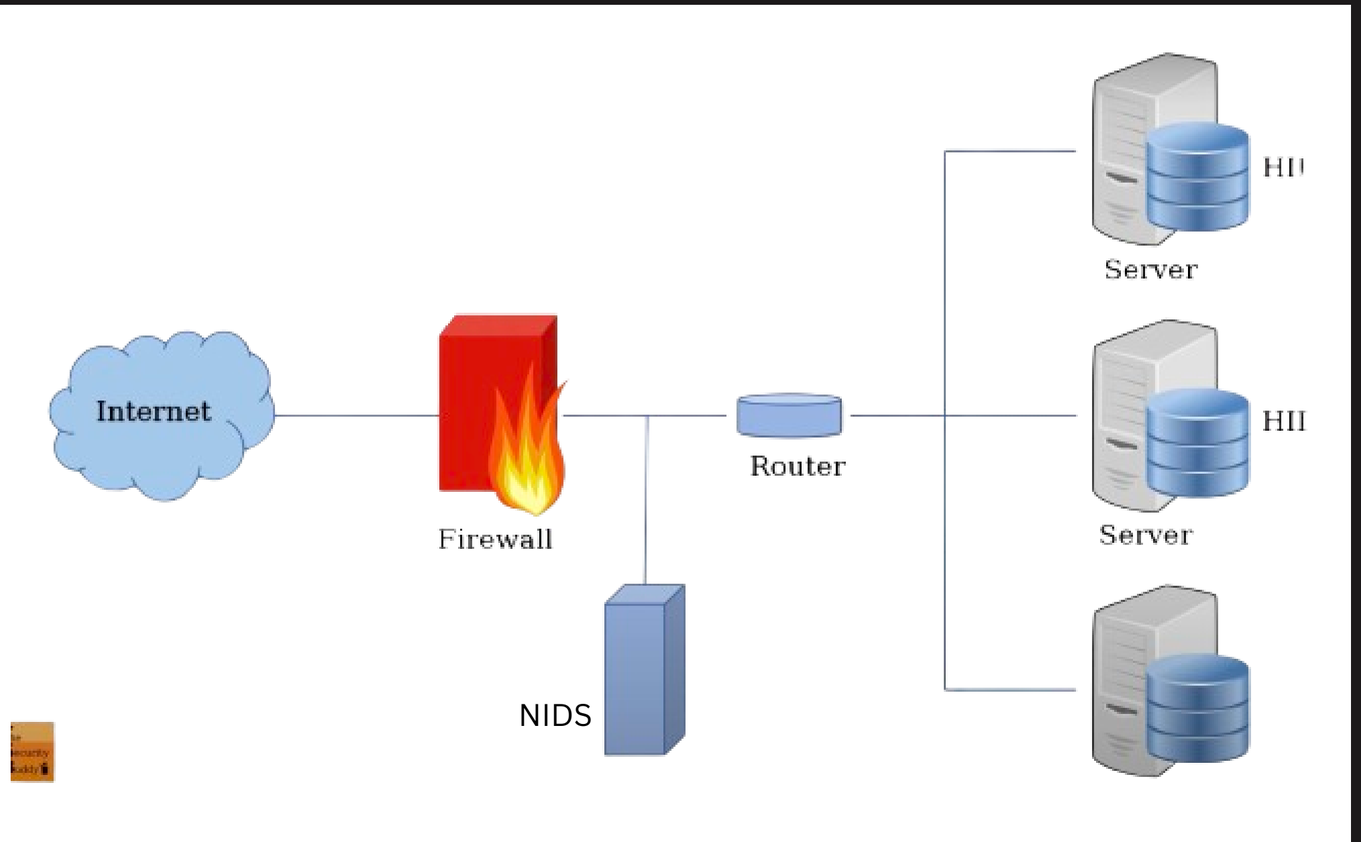
IPS(intrusion prevention system)

3.1 INTRUSION PREVENTION SYSTEMS (IPS)

These systems not only detect but also actively prevent intrusions by blocking or filtering malicious traffic.



3.2 NETWORK PROTOCOL BASED IDS

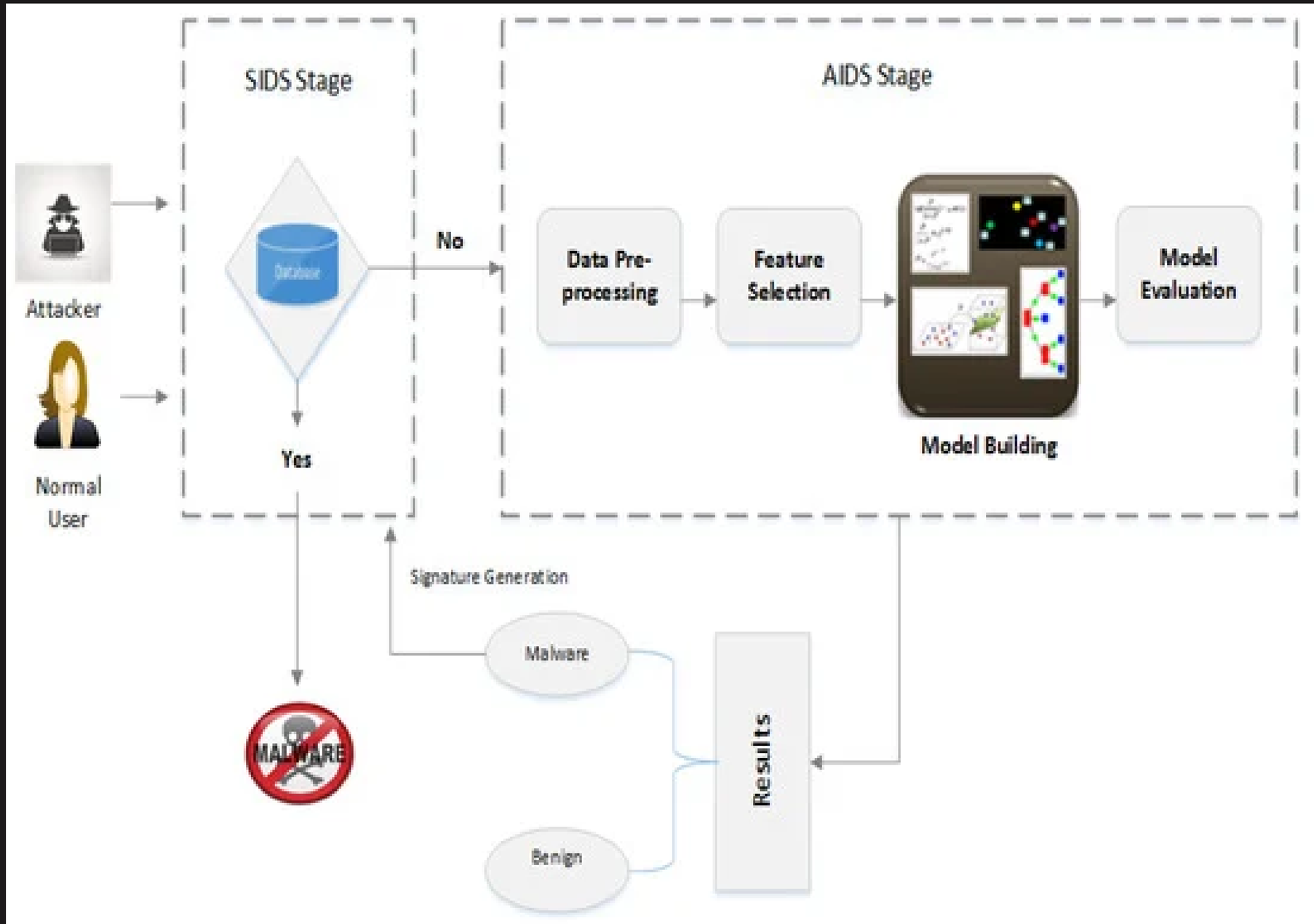


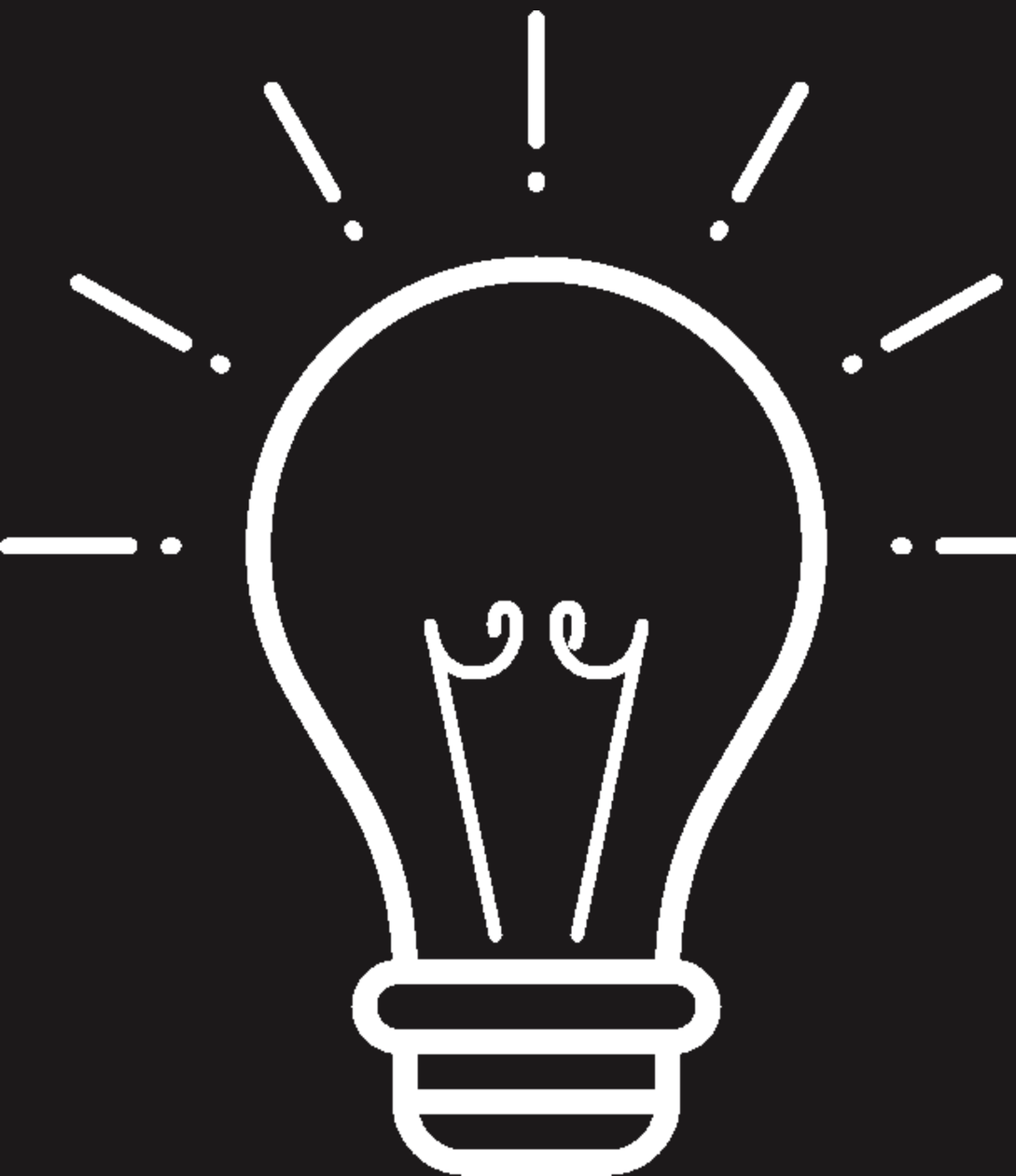
examines network traffic, protocols, and behaviors to identify and alert administrators to suspicious activities or potential security threats within the overall network.

- eg.The IDS, monitoring HTTP behavior, detects an anomaly: an HTTP request attempting to access data without going through the usual login process.

3.3 HYBRID IDS

Hybrid IDS is developed to integrates (signature)SIDS and (anomaly)AIDS to detect both unknown and known attacks.





SECTION 4: ANALYSIS SCHEMES FOR INTRUSION

4.1 ANALYSIS SCHEMES FOR INSTRUSION

- Intrusion analysis involves examining the data collected by the IDS to determine the nature and severity of detected threats.
- IDS typically employs various analysis schemes such as signature-based analysis, anomaly-based analysis, or stateful protocol analysis to identify potential intrusions.

4.2 WHY ANALYSIS SCHEMES ?

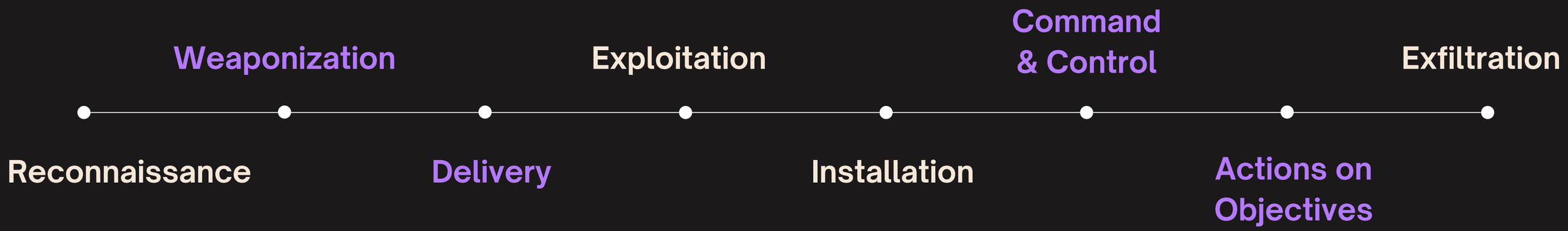
- Threat Detection
- Contextual Awareness
- Early Warning System
- Comprehensive Protection
- Investigation Aid

4.3 CYBER KILL CHAIN MODEL

- The Cyber Kill Chain model is a framework developed by Lockheed Martin to illustrate the lifecycle of a cyber attack. It provides a structured approach for understanding the various steps that an attacker typically goes through during the course of an intrusion.

- By analyzing the attack lifecycle, organizations can better detect, prevent, and respond to cyber threats, thereby enhancing their overall cybersecurity posture.

4.3 STEPS CYBER KILL CHAIN MODEL



1. Reconnaissance: Attackers gather information about the target organization, including network architecture and potential vulnerabilities.

2. Weaponization: Attackers create or obtain malware to exploit identified vulnerabilities.

3. Delivery: Malware is delivered to the target environment through various means like phishing emails or malicious websites.

4. Exploitation: Vulnerabilities within the target environment are exploited to gain unauthorized access.

5. Installation: Attackers install and execute malicious code or tools on compromised systems to establish persistence.

6. Command and Control (C2): Attackers establish communication channels with external servers to remotely manage compromised systems.

7. Actions on Objectives: Attackers carry out their intended goals, such as data theft or disruption of operations.

8. Exfiltration: Stolen data is transferred out of the target environment to complete the intrusion cycle.

SECTION 5: SECURITY ASSESSMENTS

5.1 TECHNIQUES FOR INTRUSION ANALYSIS

- Log Analysis
- Network Traffic Analysis
- Memory Forensics
- Attacker Profiling
- Endpoint Detection and Response (EDR)
- Threat Intelligence

5.2 MAPPING RESPONSES TO POLICY

- Identify relevant policies: Align incident response actions with established policies like AUP, SIRP, and data breach notification policies.
- Map policies to response actions: Create a clear matrix linking specific response actions to the policies they uphold.
- Train and communicate: Educate all personnel on policy alignment to ensure everyone understands their roles and response actions.
- Benefits:
- 1. Consistency: Uniform approach to handling incidents.
- 2. Efficiency: Streamlined response process.
- 3. Transparency: Builds trust through adherence to established policies.

5.3 VULNERABILITY ANALYSIS

- Proactive Security Measures: Highlight the proactive nature of vulnerability analysis in identifying and addressing weaknesses before they can be exploited.
- Tools and Methodologies: common tools and methodologies used for vulnerability analysis (e.g., vulnerability scanners, penetration testing).
- Risk Mitigation: Emphasize how vulnerability analysis contributes to risk mitigation and overall security enhancement.

5.4 CREDENTIAL ANALYSIS

- Authentication Security: Stress the importance of secure authentication mechanisms and the role of credential analysis in maintaining their integrity.
- Common Vulnerabilities: common vulnerabilities associated with credentials, such as weak passwords, credential reuse, and password storage practices.
- Access Control Enhancement: effective credential analysis contributes to strengthening access controls and preventing unauthorized access.

Thank you!

Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title: Design and Implementation of Diffie Hellman Key Establishment

Title: Design and Implementation of Diffie Hellman Key Establishment

---

Explanation:

1. **Setup**: Both parties agree on two public parameters:
   - A large prime number p.
   - A primitive root g modulo p, which is an integer less than p.
2. **Private Key Generation**: Both Alice and Bob independently choose secret integers:
   - Alice selects a secret integer and keeps it private.
   - Bob selects a secret integer and keeps it private.
3. **Public Key Computation**: Both parties compute their public keys:
   - Alice calculates her public key by raising the base g to the power of her secret integer modulo p.
   - Bob calculates his public key by raising the base g to the power of his secret integer modulo p.
4. **Key Exchange**: Alice and Bob exchange their public keys over the insecure channel.
5. **Shared Secret Calculation**: Once both parties receive each other's public keys:
   - Alice computes the shared secret key by raising Bob's public key to the power of her secret integer modulo p.
   - Bob computes the shared secret key by raising Alice's public key to the power of his secret integer modulo p.
6. **Result**: Both Alice and Bob now possess the same shared secret key, which they can use for encryption and decryption purposes.

The security of the Diffie-Hellman key exchange relies on the computational difficulty of solving the discrete logarithm problem, making it practically infeasible for an eavesdropper to determine the shared secret key even if they intercept the exchanged public keys.

Result:

Public Information:

Prime Number:

7237 Generate Prime

Generator G:

26 Another Generator

Alice

Key: 3842 Generate A

4482 Calculate Ga

Send Ga to B

Received:

Calculate Gab

Bob

Key: 2845 Generate B

1761 Calculate Gb

Send Gb to A

Received:

Calculate Gba

Public Information:

Prime Number:

7237 Generate Prime

Generator G:

26 Another Generator

Alice

Key: 4559 Generate A

6963 Calculate Ga

Send Ga to B

Received: 1076

Calculate Gab 5791

Bob

Key: 6430 Generate B

1076 Calculate Gb

Send Gb to A

Received: 6963

Calculate Gba 5791

Conclusion:

In conclusion, the Diffie-Hellman key establishment protocol provides a secure method for two parties to establish a shared secret key over an insecure communication channel. By leveraging mathematical principles, specifically the difficulty of solving the discrete logarithm problem, Diffie-Hellman ensures that even if adversaries intercept the exchanged public keys, they cannot feasibly determine the shared secret key without knowledge of the private keys. This shared

secret key can then be utilized for encryption, enabling secure communication between the parties. Overall, Diffie-Hellman plays a fundamental role in modern cryptography, serving as a cornerstone for secure communication protocols and encryption systems.

Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

Design and Implementation of HMAC

Explanation:

HMAC (Hash-based Message Authentication Code):

- **Definition**: HMAC is a mechanism for verifying the authenticity and integrity of a message using a cryptographic hash function and a secret key.
- **Algorithm**: HMAC involves a hash function (such as SHA-256) and a secret key shared between the sender and receiver.
- **Key Components**:
    - **Message**: The data that needs to be authenticated.
    - **Secret Key**: A shared secret known only to the sender and receiver.
    - **Hash Function**: A cryptographic hash function used to generate a fixed-size hash value from the input data.
- **Process**:

1. The sender computes a hash-based message authentication code using the message and the secret key.

2. The receiver independently computes the HMAC using the received message and the shared secret.

3. The receiver compares the computed HMAC with the received HMAC. If they match, the message is considered authentic and intact.

Advantages of HMAC:

- **Security**: HMAC provides strong security guarantees against message tampering and forgery.
- **Efficiency**: It offers efficient verification of message integrity without transmitting the entire message.
- **Flexibility**: HMAC can be implemented using various hash functions, allowing for flexibility in choosing the appropriate algorithm for the application.

Implementation:

```
#include <functional>
#include <iostream>
#include <string>
#include <vector>
using namespace std;

size_t stringHashing(string s)
{

    // Get the string
    // to get its hash value
    string hashing1 = s;

    // Instantiation of Object
    hash<string> mystdhash;

    // Using operator() to get hash value



    size_t ans=mystdhash(hashing1);
    return ans;

}

string encryption(string s){

    int n=s.size();
    string encry_s="";
    for(int i=0;i<n;i++){
        char temp=s[i]+1;
        encry_s= encry_s + (temp);
    }

    return encry_s;
}

string decryption(string s){
```

```
    int n=s.size();
    string decry_s="";
    for(int i=0;i<n;i++){
        char temp=s[i]-1;
        decry_s= decry_s + (temp);
    }

    return decry_s;
}

void manupulate(string &s){
    s[0]=s[0]+1;
}

int main(){

    string message;

    cin>>message;

    size_t hashed_message_int=stringHashing(message);
    string hashedMessage=to_string(hashed_message_int);
    string encrypt_hash=encryption(hashedMessage);
    string encrypt_message=encryption(message);

    cout<<"The sender side"<<endl;

    cout<<"The orignal message"<<endl;
    cout<<message<<endl;
    cout<<"the hashed message"<<endl;
    cout<<hashedMessage<<endl;
    cout<<"the encrypted hashed message "<<endl;
    cout<<encrypt_hash<<endl;
    cout<<"the encrypted  message "<<endl;
    cout<<encrypt_message<<endl;
    cout<<endl;
    cout<<endl;
    cout<<endl;


    cout<<"The receiver side"<<endl;
```

```
    cout<<"Do you want to manupulate the data"<<endl;
    cout<<"1=>Yes"<<endl;
    cout<<"2=>NO"<<endl;

    int t;
    cin>>t;

    if(t==1){
        manupulate(encrypt_message);
    }

    string decrypted_message= decryption(encrypt_message);
    string decrypted_hash=decryption(encrypt_hash);
    string hashed_decrypted_message=to_string(stringHashing(decrypted_message));
    cout<<"decrypted message"<<endl;
    cout<<decrypted_message<<endl;
    cout<<"decrypted hash"<<endl;
    cout<<decrypted_hash<<endl;
    cout<<"Hashed decrypted message"<<endl;
    cout<<hashed_decrypted_message<<endl;









    if(hashed_decrypted_message==decrypted_hash){
        cout<<"correct message"<<endl;
    }else{
        cout<<"incorrect message"<<endl;
    }



    return 0;
}
```

Conclusion:

In your lab work on HMAC (Hash-based Message Authentication Code), you've implemented a simple demonstration of how HMAC can be used for message integrity verification. Let's delve into some theory and then provide a conclusion you can include in your lab report.

Theory:

HMAC (Hash-based Message Authentication Code):

- **Definition**: HMAC is a mechanism for verifying the authenticity and integrity of a message using a cryptographic hash function and a secret key.
- **Algorithm**: HMAC involves a hash function (such as SHA-256) and a secret key shared between the sender and receiver.
- **Key Components**:
    - **Message**: The data that needs to be authenticated.
    - **Secret Key**: A shared secret known only to the sender and receiver.
    - **Hash Function**: A cryptographic hash function used to generate a fixed-size hash value from the input data.
- **Process**:

1. The sender computes a hash-based message authentication code using the message and the secret key.

2. The receiver independently computes the HMAC using the received message and the shared secret.

3. The receiver compares the computed HMAC with the received HMAC. If they match, the message is considered authentic and intact.

Advantages of HMAC:

- **Security**: HMAC provides strong security guarantees against message tampering and forgery.

- **Efficiency**: It offers efficient verification of message integrity without transmitting the entire message.
- **Flexibility**: HMAC can be implemented using various hash functions, allowing for flexibility in choosing the appropriate algorithm for the application.

Conclusion:

In conclusion, my implementation demonstrates the practical application of HMAC for ensuring message integrity in communication systems. By combining a hash function with a secret key, HMAC provides a reliable mechanism for verifying the authenticity of transmitted data. Through this lab work, I 've gained hands-on experience in implementing HMAC, understanding its key components, and evaluating its effectiveness in detecting message tampering. Overall, HMAC emerges as a valuable tool in maintaining the security and trustworthiness of communication protocols, offering robust protection against unauthorized alterations to transmitted information.

Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

Design and Implementation of DOS using Hping 3

Explanation:

1. **Denial of Service (DoS) Attack**:
    - In a DoS attack, a single source is used to flood a target with a large volume of traffic, thus consuming all available resources and making the service unavailable to legitimate users.
    - This attack can be launched using various methods, including sending a flood of TCP SYN packets, UDP packets, or ICMP (ping) requests to the target.
2. **Distributed Denial of Service (DDoS) Attack**:
    - DDoS attacks are more sophisticated and potent than DoS attacks as they involve multiple sources, often distributed across the internet, coordinated to flood the target simultaneously.
    - DDoS attacks typically leverage botnets, which are networks of compromised computers (often referred to as zombies) that can be remotely controlled by an attacker. These botnets are used to amplify the attack and make it more difficult to mitigate.
3. **Using hping3 for DoS/DDoS Attacks**:
    - hping3 is a command-line tool used for network testing and manipulation. It can be abused by attackers to launch DoS or DDoS attacks by sending crafted packets to a target.
    - hping3 can be used to send various types of packets, including TCP SYN, UDP, and ICMP packets, with custom payloads and rates.
    - Attackers can use hping3 to flood a target with packets, overwhelming its network bandwidth, exhausting its resources (such as CPU or memory), or exploiting vulnerabilities in the target's network stack.
    - hping3 can also be used to perform more sophisticated attacks, such as TCP SYN flooding, UDP flooding, ICMP flooding, and TCP ACK flooding.

4. **Mitigation and Prevention**:

   - Defending against DoS and DDoS attacks requires a combination of proactive measures and reactive strategies.
   - Proactive measures include implementing network security best practices, such as firewalls, intrusion detection/prevention systems (IDS/IPS), rate limiting, and filtering out malicious traffic.
   - Reactive strategies involve detecting and mitigating attacks in real-time using specialized DDoS mitigation appliances or cloud-based DDoS protection services.
   - Additionally, organizations should have incident response plans in place to quickly respond to and recover from DoS/DDoS attacks, including identifying the source of the attack and working with law enforcement if necessary.

Implementation:

DoS attack with hping3

dos.md Raw

DoS Attack with Hping3

Run the command: `hping3 --flood -S -V --rand-source http://stv.com`

Where:

1. `--flood` send packets as fast as possible
2. `-S` (Syn packet): legit TCP packet connection
3. `-V` verbose mode
4. `--rand-source` randomize the IP source address, like it's requested from different systems (sort of DDoS)

clhore commented on Feb 1, 2022

In this case the '-V' parameter is not really necessary since adding the '--flood' parameter disables it.

```
hping3: hping in flood mode, no replies will be shown
```

You can add the parameter '-d' or '--data' to be able to modify the weight of the sent data.
And if you wanted to use a single fake IP instead of multiple ones, you could use '-a' or '--spoof'.

Ej:

```
sudo hping3 --flood -S -d 2000 -a <fake_ip> <target_ip>
```

```
root@kali:~# ping -c 3 10.0.0.37
PING 10.0.0.37 (10.0.0.37) 56(84) bytes of data.
64 bytes from 10.0.0.37: icmp_seq=1 ttl=64 time=0.372 ms
64 bytes from 10.0.0.37: icmp_seq=2 ttl=64 time=0.236 ms
64 bytes from 10.0.0.37: icmp_seq=3 ttl=64 time=0.218 ms

--- 10.0.0.37 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.218/0.275/0.372/0.070 ms
root@kali:~#
```

```
root@kali:~# hping3 -S --flood --interface wlan0 --rand-source 10.0.0.37
```

```
root@kali:~# hping3 -S --flood --interface wlan0 --rand-source 10.0.0.37
HPING 10.0.0.37 (wlan0 10.0.0.37): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

246.99.62.66	10.0.0.37	TCP	54	1825→0 [SYN] Seq=0 Win=512 Len=0
152.246.145.17	10.0.0.37	TCP	54	1826→0 [SYN] Seq=0 Win=512 Len=0
17.160.192.51	10.0.0.37	TCP	54	1827→0 [SYN] Seq=0 Win=512 Len=0
217.195.51.84	10.0.0.37	TCP	54	1828→0 [SYN] Seq=0 Win=512 Len=0
1.86.43.188	10.0.0.37	TCP	54	1829→0 [SYN] Seq=0 Win=512 Len=0

Conclusion:

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are serious threats that disrupt online services by overwhelming their resources with malicious traffic. DoS attacks originate from a single source, while DDoS attacks involve multiple distributed sources, making them more potent and difficult to mitigate. Tools like hping3 can be misused by attackers to execute these attacks, although such actions are illegal and unethical. To defend against DoS and DDoS attacks, organizations should implement proactive measures like firewalls, intrusion detection/prevention systems, and rate limiting. Reactive strategies, such as real-time attack detection and mitigation, are also essential for minimizing the impact of attacks. Collaboration between network administrators, security professionals, and law enforcement agencies is crucial for identifying and prosecuting malicious actors. Ultimately, building resilient and secure networks is vital for safeguarding against the disruptive effects of DoS and DDoS attacks.

Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

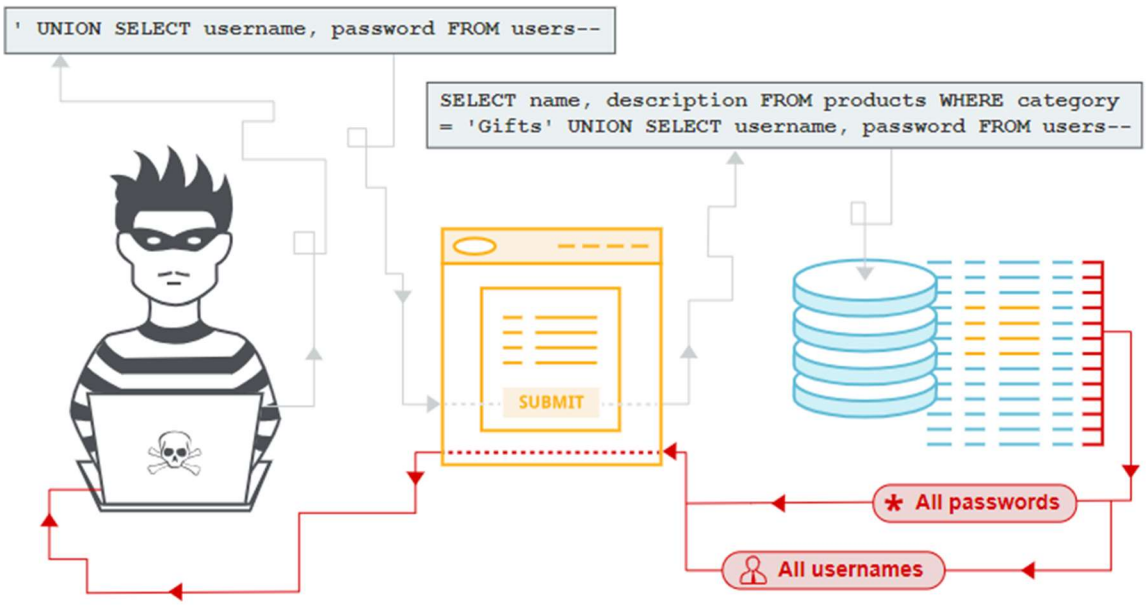
Title:

Simulation of SQL Injection (PBLE-1)

Explanation:

SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. This can allow an attacker to view data that they are not normally able to retrieve. This might include data that belongs to other users, or any other data that the application can access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

In some situations, an attacker can escalate a SQL injection attack to compromise the underlying server or other back-end infrastructure. It can also enable them to perform denial-of-service attacks.



Implementation:

1.Retrieving hidden data

Imagine a shopping application that displays products in different categories. When the user clicks on the **Gifts** category, their browser requests the URL:
https://insecure-website.com/products?category=Gifts
This causes the application to make a SQL query to retrieve details of the relevant products from the database:
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
This SQL query asks the database to return:

- all details (*)
- from the products table
- where the category is Gifts
- and released is 1.

The restriction released = 1 is being used to hide products that are not released. We could assume for unreleased products, released = 0.
The application doesn't implement any defenses against SQL injection attacks. This means an attacker can construct the following attack, for example:
https://insecure-website.com/products?category=Gifts'--
This results in the SQL query:
SELECT * FROM products WHERE category = 'Gifts'--' AND released = 1
Crucially, note that -- is a comment indicator in SQL. This means that the rest of the query is interpreted as a comment, effectively removing it. In this example, this means the query no longer includes AND released = 1. As a result, all products are displayed, including those that are not yet released.
You can use a similar attack to cause the application to display all the products in any category, including categories that they don't know about:
https://insecure-website.com/products?category=Gifts'+OR+1=1--
This results in the SQL query:
SELECT * FROM products WHERE category = 'Gifts' OR 1=1--' AND released = 1
The modified query returns all items where either the category is Gifts, or 1 is equal to 1.
As 1=1 is always true, the query returns all items.

Lab: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

APPRENTICE

LAB | Not solved

This lab contains a SQL injection vulnerability in the product category filter. When the user selects a category, the application carries out a SQL query like the following:

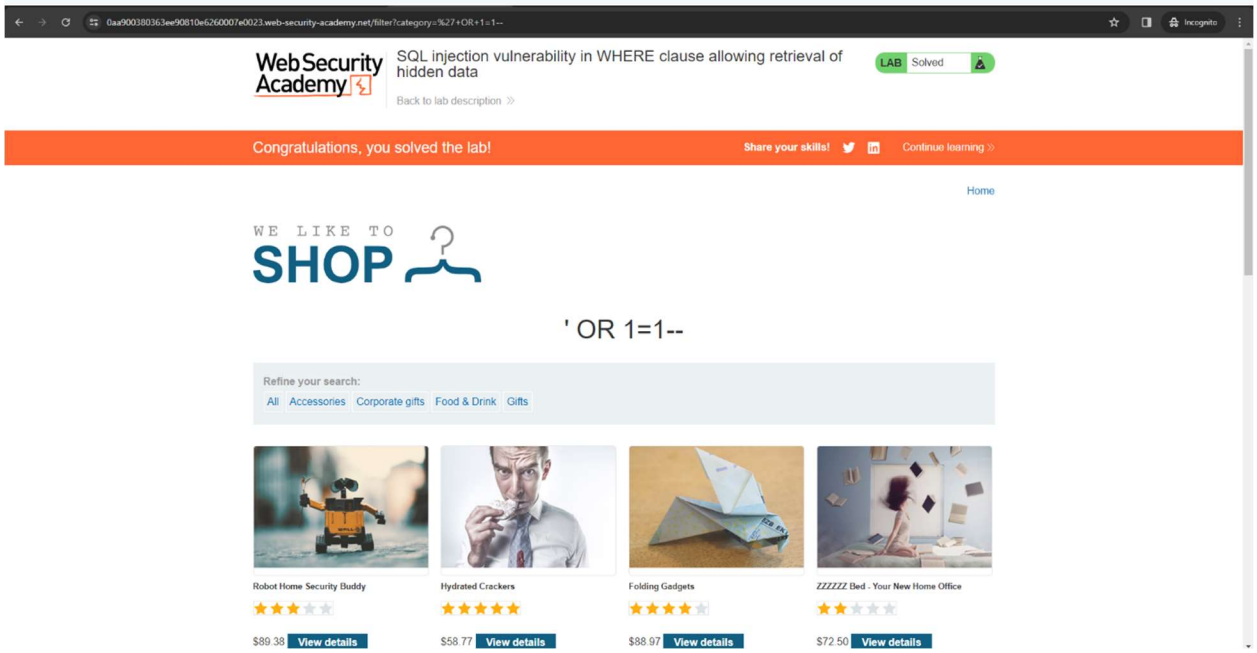
```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

To solve the lab, perform a SQL injection attack that causes the application to display one or more unreleased products.

ACCESS THE LAB

Solution

Community solutions



2. Subverting application logic

Imagine an application that lets users log in with a username and password. If a user submits the username wiener and the password bluecheese, the application checks the credentials by performing the following SQL query:

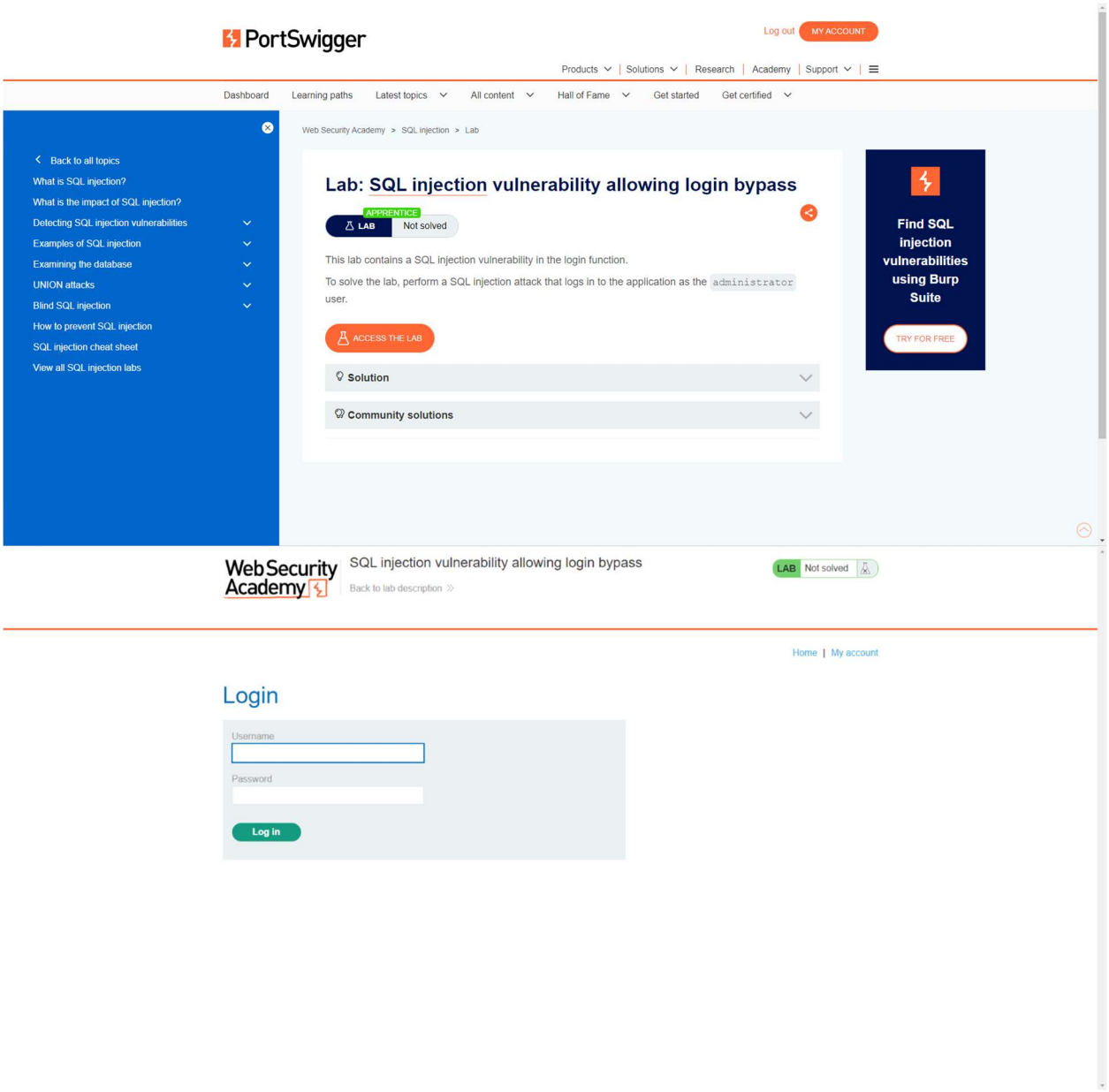
SELECT * FROM users WHERE username = 'wiener' AND password = 'bluecheese'

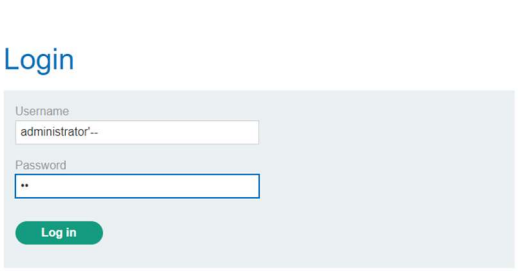
If the query returns the details of a user, then the login is successful. Otherwise, it is rejected.

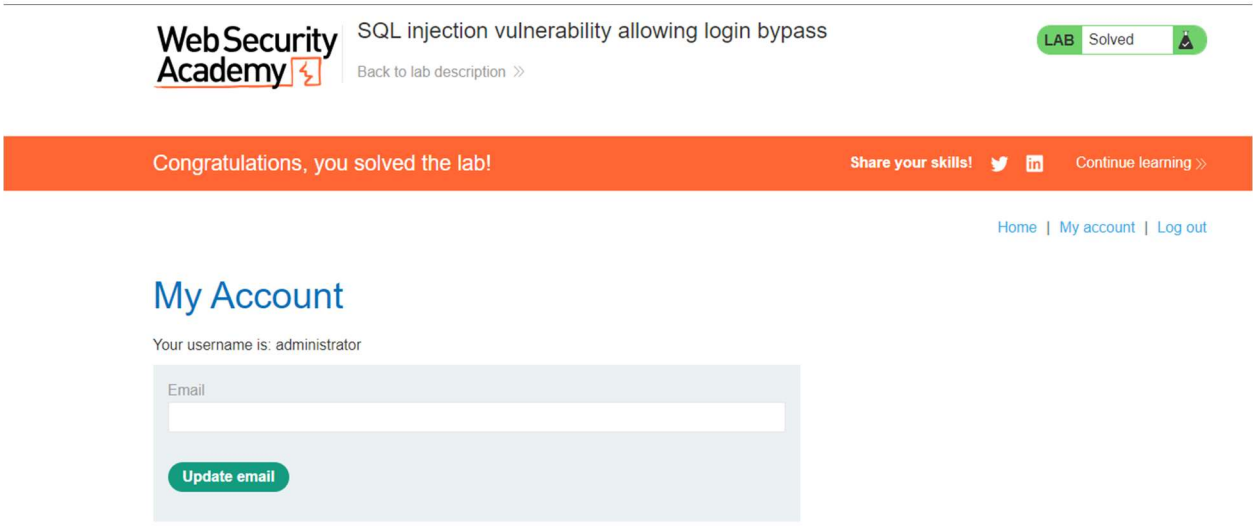
In this case, an attacker can log in as any user without the need for a password. They can do this using the SQL comment sequence -- to remove the password check from the WHERE clause of the query. For example, submitting the username administrator'-- and a blank password results in the following query:

SELECT * FROM users WHERE username = 'administrator'--' AND password = ''

This query returns the user whose username is administrator and successfully logs the attacker in


PortSwigger
Log out
MY ACCOUNT
Products | Solutions | Research | Academy | Support
Dashboard
Learning paths
Latest topics
All content
Hall of Fame
Get started
Get certified
Back to all topics
What is SQL injection?
What is the impact of SQL injection?
Detecting SQL injection vulnerabilities
Examples of SQL injection
Examining the database
UNION attacks
Blind SQL injection
How to prevent SQL injection
SQL injection cheat sheet
View all SQL injection labs
Web Security Academy > SQL injection > Lab
Lab: SQL injection vulnerability allowing login bypass
APPRENTICE
LAB
Not solved
This lab contains a SQL injection vulnerability in the login function.
To solve the lab, perform a SQL injection attack that logs in to the application as the administrator user.
ACCESS THE LAB
Solution
Community solutions
Find SQL injection vulnerabilities using Burp Suite
TRY FOR FREE
Web Security Academy
SQL injection vulnerability allowing login bypass
Back to lab description
LAB Not solved
Home | My account
Login
Username
Password
Log in


Login
Username
administrator'--
Password
Log in


Web Security Academy
SQL injection vulnerability allowing login bypass
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home | My account | Log out
My Account
Your username is: administrator
Email
Update email

3. Retrieving data from other database tables

In cases where the application responds with the results of a SQL query, an attacker can use a SQL injection vulnerability to retrieve data from other tables within the database. You can use the UNION keyword to execute an additional SELECT query and append the results to the original query.

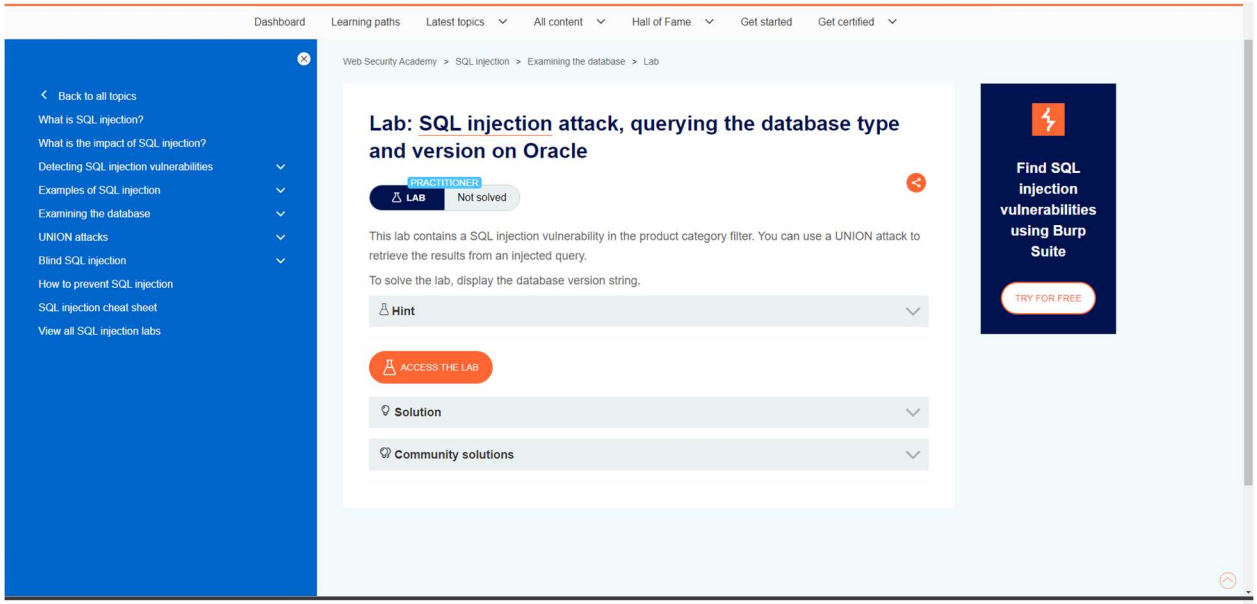
For example, if an application executes the following query containing the user input Gifts:

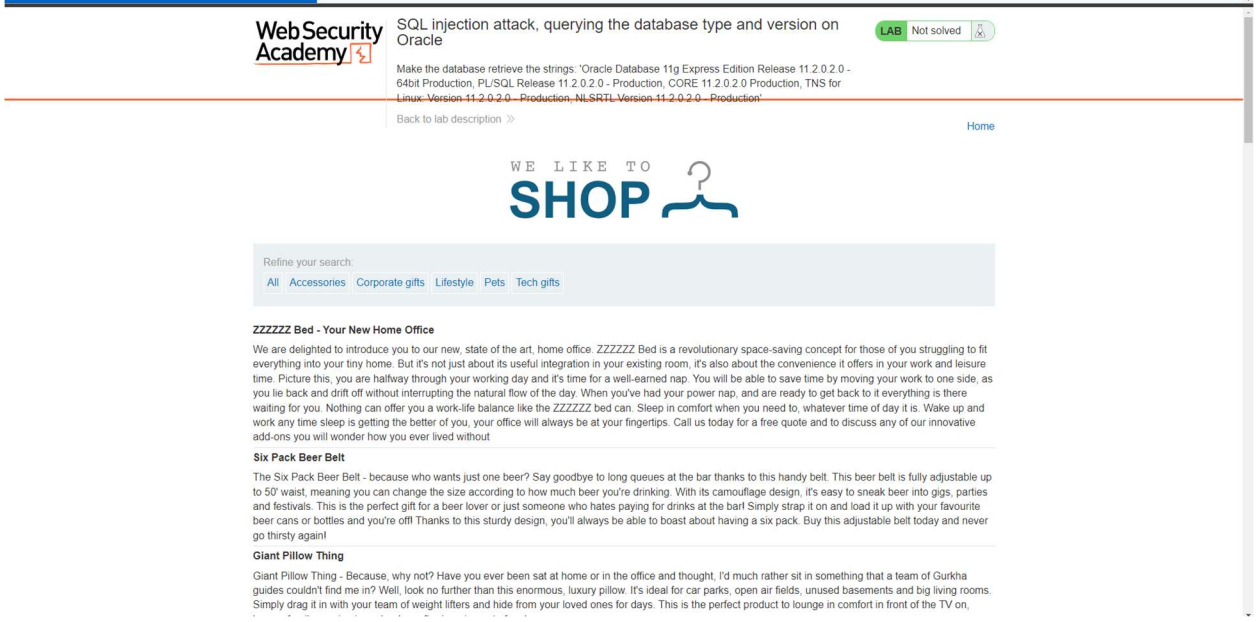
SELECT name, description FROM products WHERE category = 'Gifts'

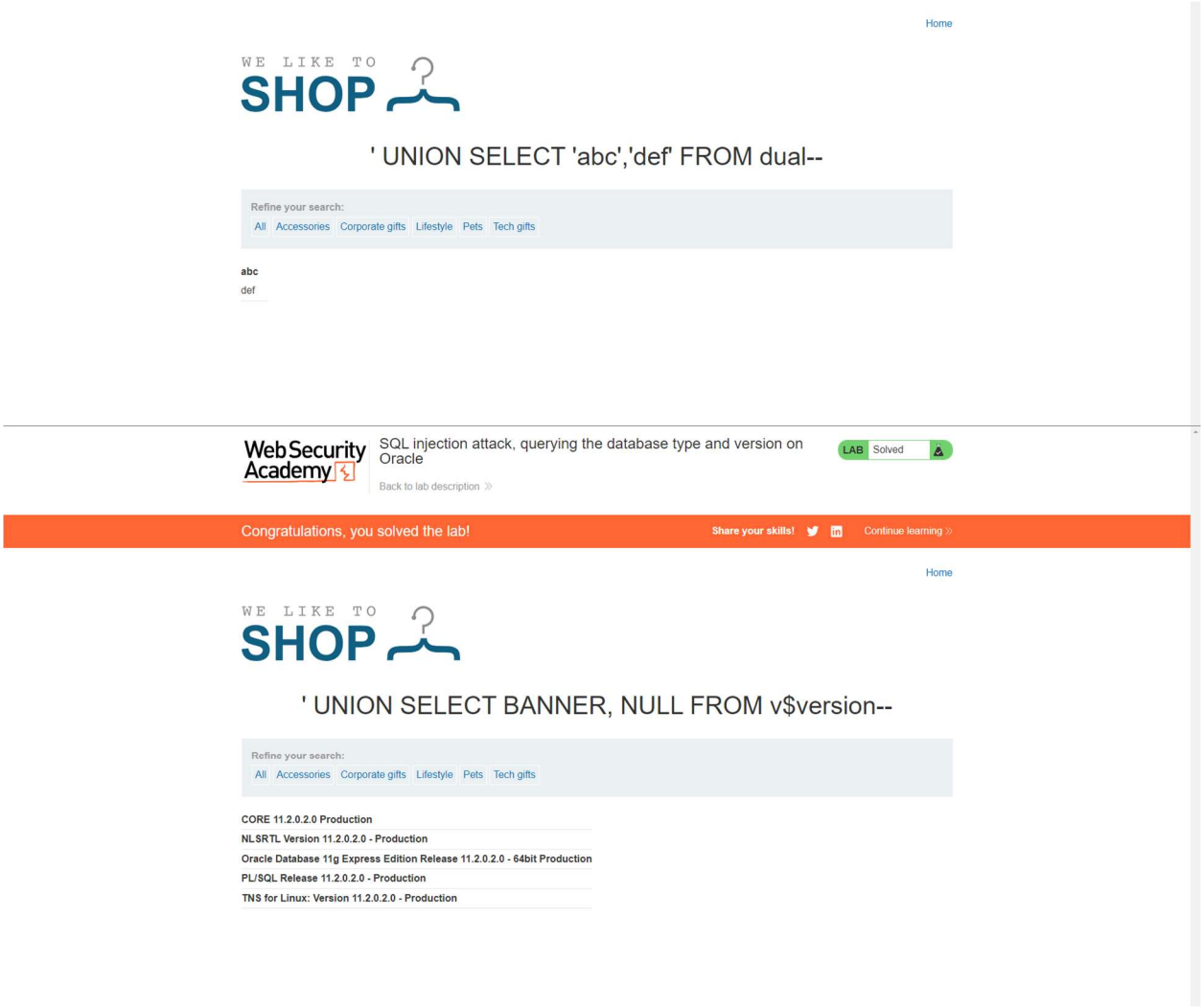
An attacker can submit the input:

' UNION SELECT username, password FROM users--

This causes the application to return all usernames and passwords along with the names and descriptions of products.


Dashboard
Learning paths
Latest topics
All content
Hall of Fame
Get started
Get certified
Back to all topics
What is SQL injection?
What is the impact of SQL injection?
Detecting SQL injection vulnerabilities
Examples of SQL injection
Examining the database
UNION attacks
Blind SQL injection
How to prevent SQL injection
SQL injection cheat sheet
View all SQL injection labs
Web Security Academy > SQL injection > Examining the database > Lab
Lab: SQL injection attack, querying the database type and version on Oracle
PRACTITIONER
LAB
Not solved
This lab contains a SQL injection vulnerability in the product category filter. You can use a UNION attack to retrieve the results from an injected query.
To solve the lab, display the database version string.
Hint
ACCESS THE LAB
Solution
Community solutions
Find SQL injection vulnerabilities using Burp Suite
TRY FOR FREE


Web Security Academy
SQL injection attack, querying the database type and version on Oracle
LAB
Not solved
Make the database retrieve the strings: 'Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production, PL/SQL Release 11.2.0.2.0 - Production, CORE 11.2.0.2.0 Production, TNS for Linux: Version 11.2.0.2.0 - Production, NLSRTL Version 11.2.0.2.0 - Production'
Back to lab description
Home
WE LIKE TO SHOP
Refine your search:
All
Accessories
Corporate gifts
Lifestyle
Pets
Tech gifts
ZZZZZZ Bed - Your New Home Office
We are delighted to introduce you to our new, state of the art, home office. ZZZZZZ Bed is a revolutionary space-saving concept for those of you struggling to fit everything into your tiny home. But it's not just about its useful integration in your existing room, it's also about the convenience it offers in your work and leisure time. Picture this, you are halfway through your working day and it's time for a well-earned nap. You will be able to save time by moving your work to one side, as you lie back and drift off without interrupting the natural flow of the day. When you've had your power nap, and are ready to get back to it everything is there waiting for you. Nothing can offer you a work-life balance like the ZZZZZZ bed can. Sleep in comfort when you need to, whatever time of day it is. Wake up and work any time sleep is getting the better of you, your office will always be at your fingertips. Call us today for a free quote and to discuss any of our innovative add-ons you will wonder how you ever lived without
Six Pack Beer Belt
The Six Pack Beer Belt - because who wants just one beer? Say goodbye to long queues at the bar thanks to this handy belt. This beer belt is fully adjustable up to 50' waist, meaning you can change the size according to how much beer you're drinking. With its camouflage design, it's easy to sneak beer into gigs, parties and festivals. This is the perfect gift for a beer lover or just someone who hates paying for drinks at the bar! Simply strap it on and load it up with your favourite beer cans or bottles and you're off! Thanks to this sturdy design, you'll always be able to boast about having a six pack. Buy this adjustable belt today and never go thirsty again!
Giant Pillow Thing
Giant Pillow Thing - Because, why not? Have you ever been sat at home or in the office and thought, I'd much rather sit in something that a team of Gurkha guides couldn't find me in? Well, look no further than this enormous, luxury pillow. It's ideal for car parks, open air fields, unused basements and big living rooms. Simply drag it in with your team of weight lifters and hide from your loved ones for days. This is the perfect product to lounge in comfort in front of the TV on,


Home
WE LIKE TO SHOP
' UNION SELECT 'abc','def' FROM dual--
Refine your search:
All Accessories Corporate gifts Lifestyle Pets Tech gifts
abc
def
Web Security Academy
SQL injection attack, querying the database type and version on Oracle
LAB Solved
Back to lab description
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home
WE LIKE TO SHOP
' UNION SELECT BANNER, NULL FROM v$version--
Refine your search:
All Accessories Corporate gifts Lifestyle Pets Tech gifts
CORE 11.2.0.2.0 Production
NLSRTL Version 11.2.0.2.0 - Production
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
PL/SQL Release 11.2.0.2.0 - Production
TNS for Linux: Version 11.2.0.2.0 - Production

Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title: One of the most famous intrusion detection system and has been known for its flexibility with different environments. You can even integrate it with Kibana and elastic cloud. PBLE-2

Title:

One of the most famous intrusion detection system and has been known for its flexibility with different environments. You can even integrate it with Kibana and elastic cloud. PBLE-2

Explanation:

Kibana and Elastic Cloud are two components of the Elastic Stack, which is a collection of open-source software products designed for search, analysis, visualization, and logging of data. Let's delve into each component in detail:

1. **Kibana**:
    - Kibana is a powerful data visualization and exploration tool that works seamlessly with Elasticsearch, providing users with an interface to interact with their data.
    - Key features of Kibana include:
        - Data Visualization: Kibana allows users to create various visualizations such as charts, graphs, maps, and histograms to represent their data in meaningful ways.
        - Dashboard Creation: Users can combine multiple visualizations into interactive dashboards, enabling them to monitor key metrics and trends at a glance.
        - Search and Explore: Kibana provides a search interface for querying data stored in Elasticsearch, allowing users to explore their data in real-time.
        - Advanced Analytics: Kibana supports advanced analytics features like machine learning, anomaly detection, and forecasting for gaining insights from data.
        - Plugins and Extensions: Kibana's plugin architecture allows for customization and integration with third-party tools and services, expanding its functionality.
2. **Elastic Cloud**:

- Elastic Cloud is a fully managed cloud service offered by Elastic, providing users with a hassle-free way to deploy, manage, and scale Elasticsearch clusters in the cloud.
- Key features of Elastic Cloud include:
    - Managed Infrastructure: Elastic Cloud handles the underlying infrastructure, including provisioning, scaling, monitoring, and maintenance of Elasticsearch clusters, allowing users to focus on their data and applications.
    - Seamless Integration: Elastic Cloud seamlessly integrates with other components of the Elastic Stack, such as Kibana, Beats, and Logstash, providing a unified platform for data ingestion, analysis, and visualization.
    - Security and Compliance: Elastic Cloud offers built-in security features like encryption, role-based access control (RBAC), and compliance certifications (e.g., SOC 2, HIPAA) to ensure the confidentiality, integrity, and availability of data.
    - High Availability: Elastic Cloud ensures high availability and fault tolerance by automatically distributing data across multiple nodes and availability zones within the cloud provider's infrastructure.
    - Elastic Support: Elastic Cloud subscribers benefit from access to Elastic's support services, including technical support, training resources, and community forums, for assistance with troubleshooting, optimization, and best practices.

Implementation:

Title: One of the most famous intrusion detection system and has been known for its flexibility with different environments. You can even integrate it with Kibana and elastic cloud. PBLE-2

Conclusion:

Overall, Kibana's integration with Elasticsearch, real-time data visualization capabilities, ease of use, extensibility, scalability, and support for diverse data sources make it a compelling choice for organizations seeking to derive insights from their data effectively. While traditional analytical tools may offer similar features, Kibana's unique combination of functionality and usability sets it apart in the realm of data visualization and analytics.