HONEYWORDS

MAKING PASSWORD-CRACKING DETECTABLE

Jessica Collins
44345956

PASSWORD PROTECTION

USERS

Chosen passwords must be hard-to-guess, not shared, and only transmitted over encrypted channels

SERVERS

Should not store passwords in cleartext, but should keep them hashed in a password file

CREDENTIAL THEFT

The time taken between the theft and detection puts off the application of countermeasures to limit damage

CONTENTS OF THIS SEMINAR

01

HASHED PASSWORDS

Issues with security and privacy

02

HONEYWORDS

Password-cracking detection

03

SECURITY ANALYSIS

Identification of possible shortcomings

04

DISCUSSION

Evaluation of the honeyword system

PASSWORD HASHING

Used to verify the integrity of a submitted password against its stored hash to prevent actual passwords being stored by servers

PASSWORD HASHING

P = “hello”

H(P) = 3d3929g23994939e83b2ac5b9e29e1b1c1384

P′ = “hbllo”

H(P’) = 8dfac912a93f8169afe7dd238f33644939e83b

H(P’) ≠ H(P) → Login attempt is rejected

PASSWORD SALTING

Used to force a hash's uniqueness, increase its complexity without increasing user requirements, and to mitigate against password attacks

PASSWORD SALTING

User → $(s, H(s, P))$

P = “hello”

H(P) = a90219323994939e83b2ac5b9e29e1b1c19384

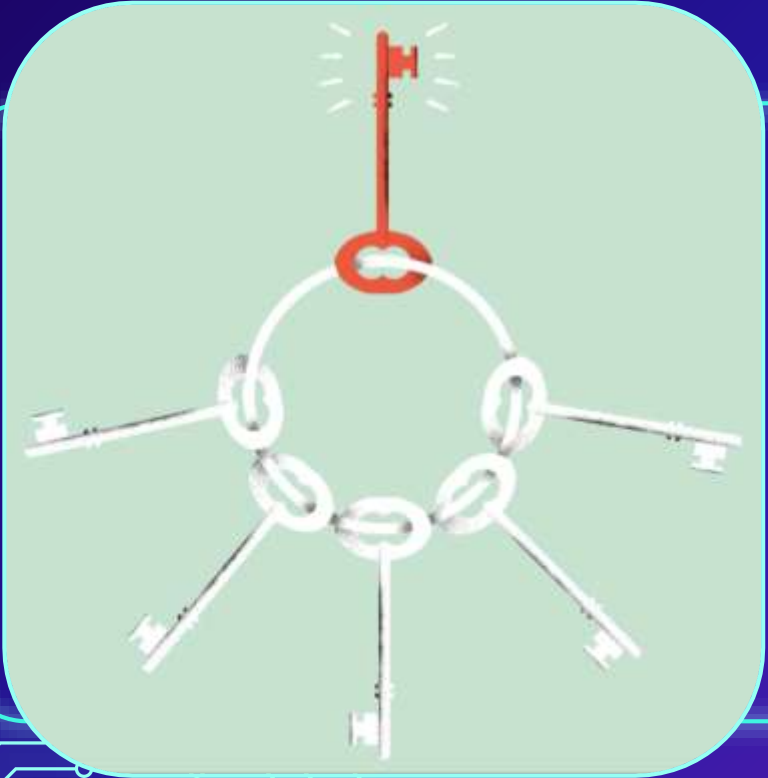
H(P + “Qxe39dfkdx”) = 8dfac912a93f8as98d8sd09sd9s3644939e3b1

H(P + “S399d3x94d”) = c9d9d9s7dd38f3364493938f33644939d3fg4f

$H(s, P') = H(s, P)$ → Login attempt is accepted

EXISTING CHALLENGES

Users adopt passwords with a poor bitstrength, reuse or forget their passwords and are susceptible to phishing attacks



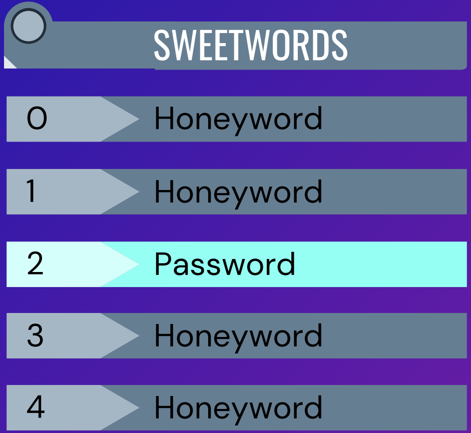
HONEYWORDS SYSTEM

Juels & Rivest, 2013

ACM Conference on Computer and Communications Security

HONEYWORDS SYSTEM

- Hash password is stored and hidden amongst a list of decoys (honeywords)
- Honeywords are indistinguishable from the password and cannot be guessed by mistake
- Log ins using a honeyword are flagged and a contingency plan is initiated



	SWEETWORDS
0	Honeyword
1	Honeyword
2	Password
3	Honeyword
4	Honeyword

SYSTEM ARCHITECTURE

LOGIN SERVER

Keeps an ordered list of sweetwords $[h(w_x)]_u$, $x \in [1, k]$

HONEYCHECKER

Stores the index of the user’s password within the list of sweetwords

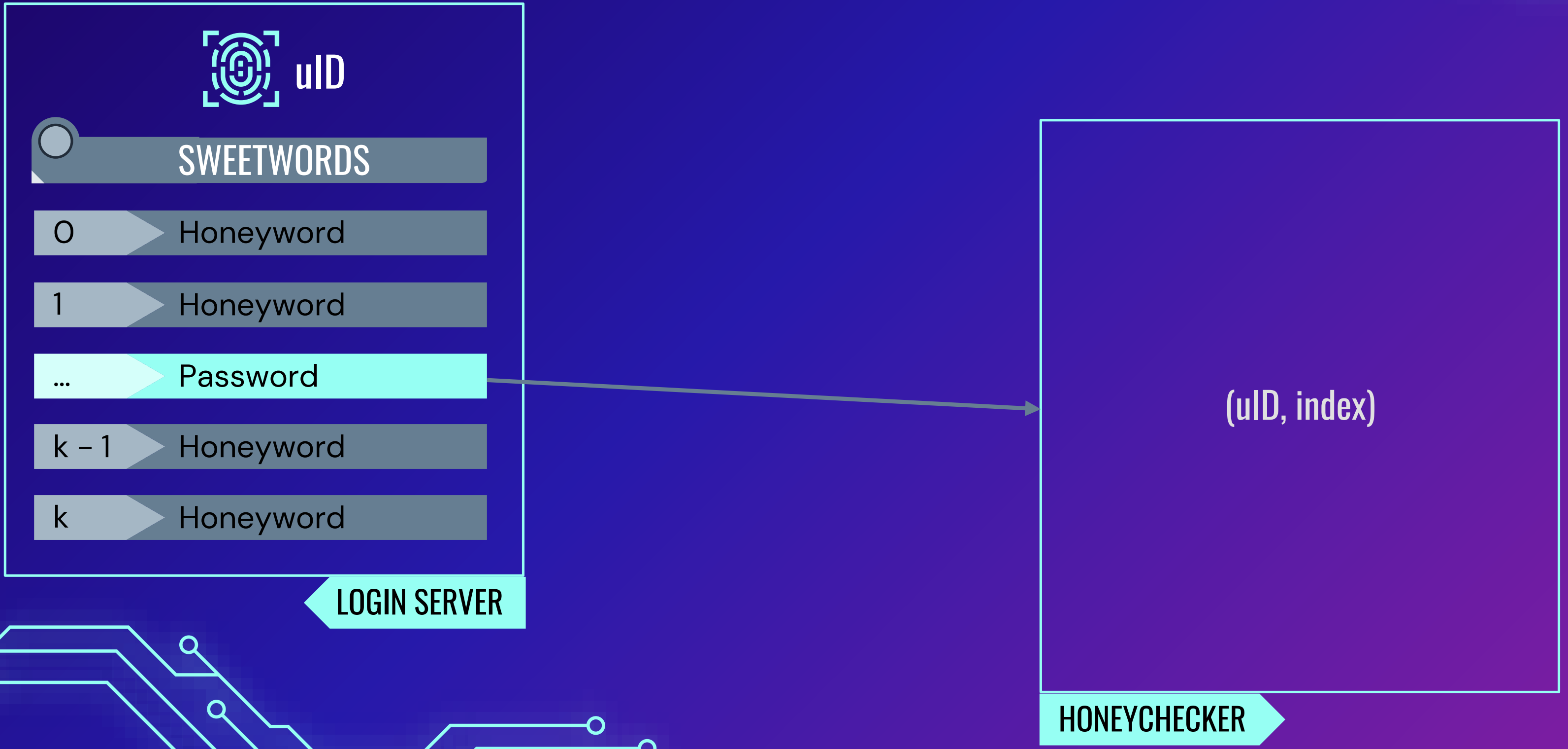
SYSTEM BEHAVIOUR

01 SETUP

02 AUTHENTICATION

03 CHANGE OF PASSWORD

SETUP PHASE



uID

SWEETWORDS

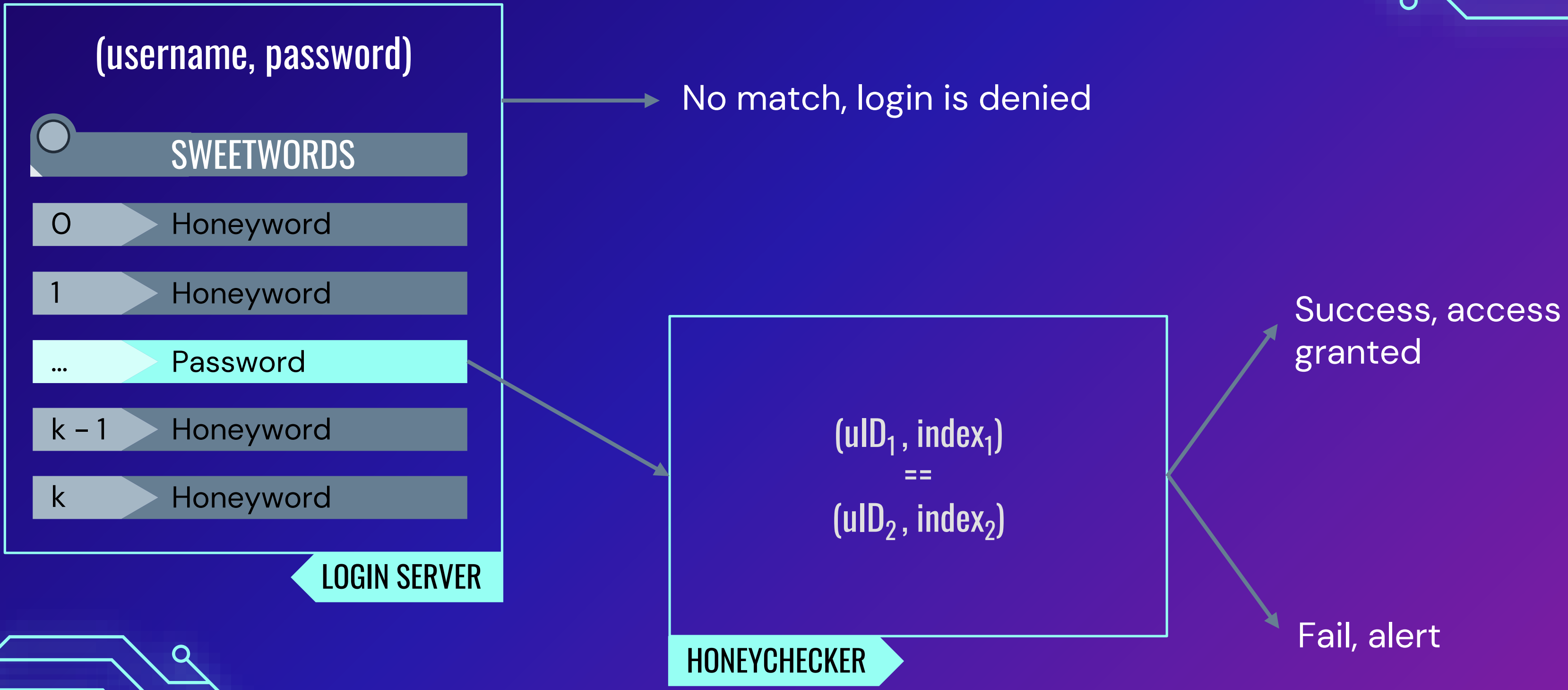
0	Honeyword
1	Honeyword
...	Password
k – 1	Honeyword
k	Honeyword

LOGIN SERVER

(uID, index)

HONEYCHECKER

AUTHENTICATION PHASE



(username, password)

SWEETWORDS

0	Honeyword
1	Honeyword
...	Password
k – 1	Honeyword
k	Honeyword

LOGIN SERVER

No match, login is denied

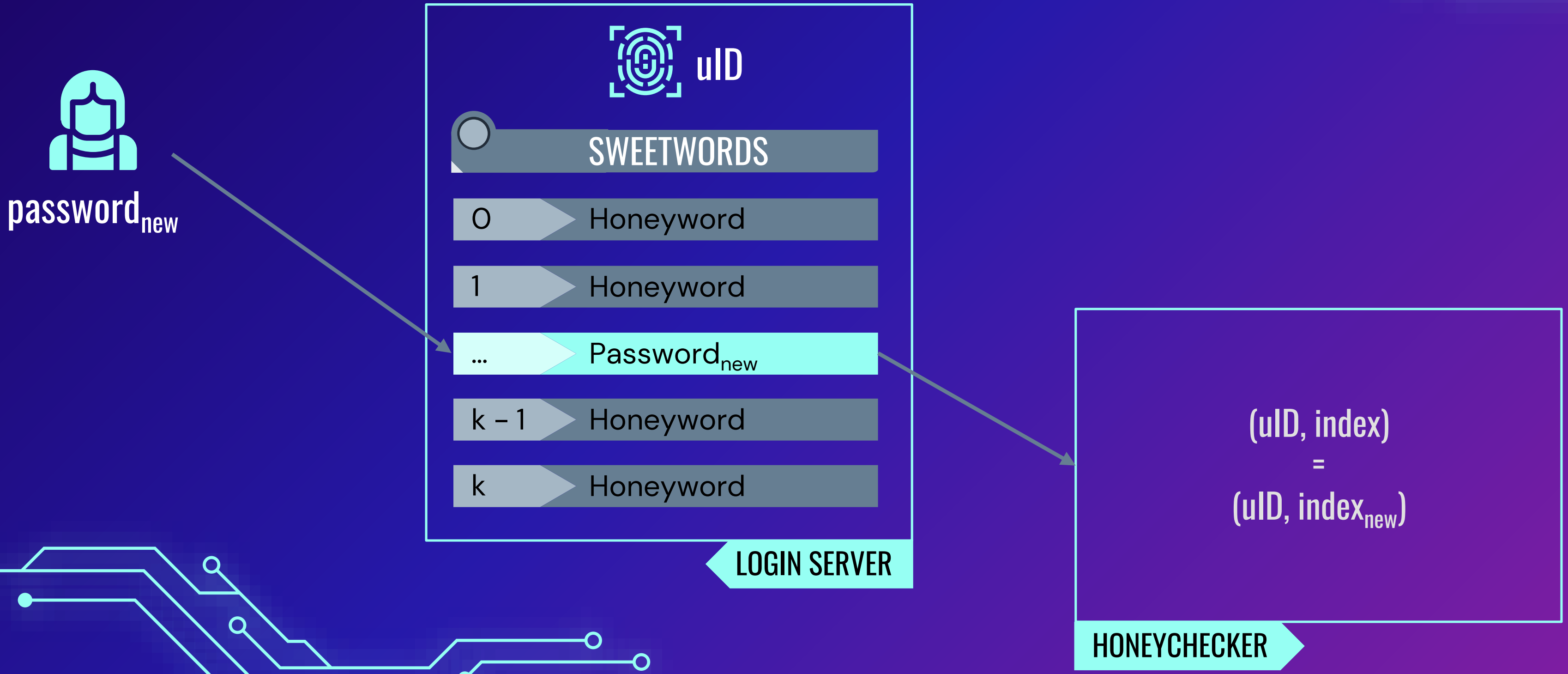
$$(uID_1\,,\ index_1) \\ == \\ (uID_2\,,\ index_2)$$

HONEYCHECKER

Success, access granted

Fail, alert

CHANGE OF PASSWORD PHASE



password$_{new}$

uID

SWEETWORDS	
0	Honeyword
1	Honeyword
...	Password$_{new}$
k – 1	Honeyword
k	Honeyword

LOGIN SERVER

(uID, index)
=
(uID, index$_{new}$)

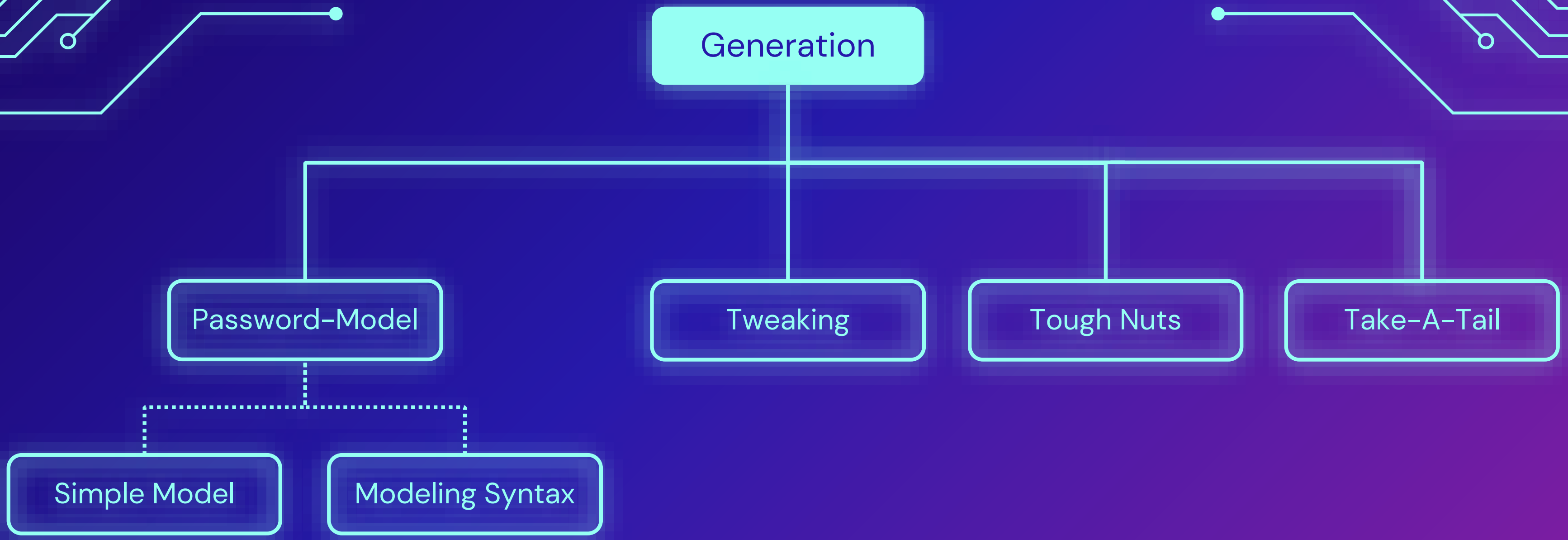
HONEYCHECKER

INTRUSION SUCCESS

- An intruder who has retrieved the sweetwords can succeed in guessing the correct password
- Probability of failure is equal to:

$$\frac{k-1}{k}$$

HONEYWORD GENERATION



Generation

Password-Model

Tweaking

Tough Nuts

Take-A-Tail

Simple Model

Modeling Syntax

GENERATION: TWEAKING

- “Tweak” selected character positions of the password to obtain honeywords
- Characters in the selected *t* positions are replaced by randomly-selected characters of the same type
- Tail-Tweaking → Replacing the characters at the tail of the password

Supplied Password: BG+7y45

Generated Sweetwords:

- BG+7q03
- BG+7m55
- BG+7y45
- BG+7o92

GENERATION: PASSWORD-MODEL

- Honeywords are generated from a list of thousands or millions of passwords
- Does not need to know the password to generate honeywords
- Modelling Syntax → Generating honeywords with each character being of the same type as the original password

Supplied Password: Unknown

Generated Honeywords:

- kebrton1
- 02123dia
- 9,5OPEe]KV.O?RIOtc&L–:IJ”b+!NWT
- forlinux
- pizzhemix01
- ‘Sb123

GENERATION: TOUGH NUTS

- Much harder to crack than average honeywords
- May never be cracked by an intruder
- May be long hashes, e.g. 256-bit random bitstring
- Can also be uncracked hashes with correct password hidden amongst them

Supplied Password: Unknown

Generated Sweetwords:

- gt79
- tom@yahoo
- ?
- 3d3929g{3],994939e83b2!nd/”8s
- rabig/30frogs!
- ?

GENERATION: TAKE-A-TAIL

- Identical to tweaking method, but new password is chosen by the user before honeywords are generated
- Randomly-selected characters are appended to user-supplied password
- Increased memorability
- Generation of honeywords is perfectly flat

```
Enter a new password: RedEye2

Append ‘413’ to make a new password.

Enter your new password: RedEye2413
```

GENERATION: HYBRIDS

- Combine the benefits from different honeyword generation strategies
- E.g. password-model and tweaking

abacad513
abacad941
abacad004
abacad752

snurfle672
snurfle806
snurfle772
snurfle091

zinja750
zinja802
zinja116
zinja649

SECURITY ANALYSIS

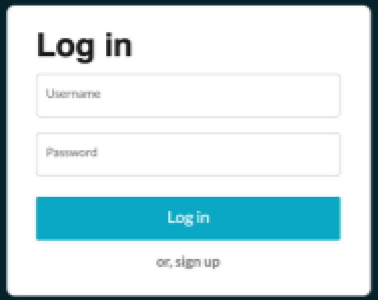
Dictionary and Brute-Force Attacks
Targeted Guessing Attacks
Denial-of-Service Attacks

DICTIONARY / BRUTE-FORCE

- The trial and error of different passwords will be detected by the Honeyword System
- It is probable that a honeyword will be used before the actual password is given
- Once the login tolerance is reached, a notification will be sent to the user or their account will be deactivated

TARGETED GUESSING

- Knowledge of a user’s information enables intruders to crack their hashed password more easily
- Password-model and tough nuts → Honeywords are easily distinguishable from user’s password
- Tweaking and take-a-tail → Honeywords differ from user’s password by only a few characters



DENIAL-OF-SERVICE

- Honeywords can be submitted to produce a false negative feedback signal
- Web server may be blocked to reduce DoS potency
- Password-model and tough nuts → Honeywords can be identified and submitted to system
- Tweaking and take-a-tail → Honeywords are difficult to distinguish from real password

SECURITY ANALYSIS: SUMMARY

	DICTIONARY / BRUTE-FORCE RESISTANCE	TARGETED GUESSING RESISTANCE	DENIAL-OF-SERVICE RESISTANCE
TWEAKING	Strong	Strong	Weak
PASSWORD-MODEL	Strong	Weak	Strong
TOUGH NUTS	Strong	Weak	Strong
TAKE-A-TAIL	Strong	Strong	Weak

DISCUSSION

Strengths and limitations of the Honeyword System

STRENGTHS

FALSE LOGIN ATTEMPTS

Detected quickly to activate countermeasures

ONE-SIDED COMPROMISE

HC and LS are run separately to prevent mutual compromise

ADMINISTRATIVE EFFORT

Just have to wait for password breaches to occur

HACKER CONFIDENCE

A false successful login does not mean it hasn’t been detected

WEAKNESSES

CO-RELATIONAL HAZARD

Relationships between usernames and passwords prevent honeywords from protecting the original password

DISTINGUISHABLE PASSWORD PATTERNS

Well-known password patterns can be recognised from a list of sweetwords

DENIAL-OF-SERVICE RESISTIVITY

If a user's passwords are known, the accompanying honeywords can be used to execute a DoS attack

MULTIPLE SYSTEM VULNERABILITY

The use of a password across several systems employing the same honeyword generator can bring about MSV

HONEYWORDS

MAKING PASSWORD-CRACKING DETECTABLE

Jessica Collins
44345956