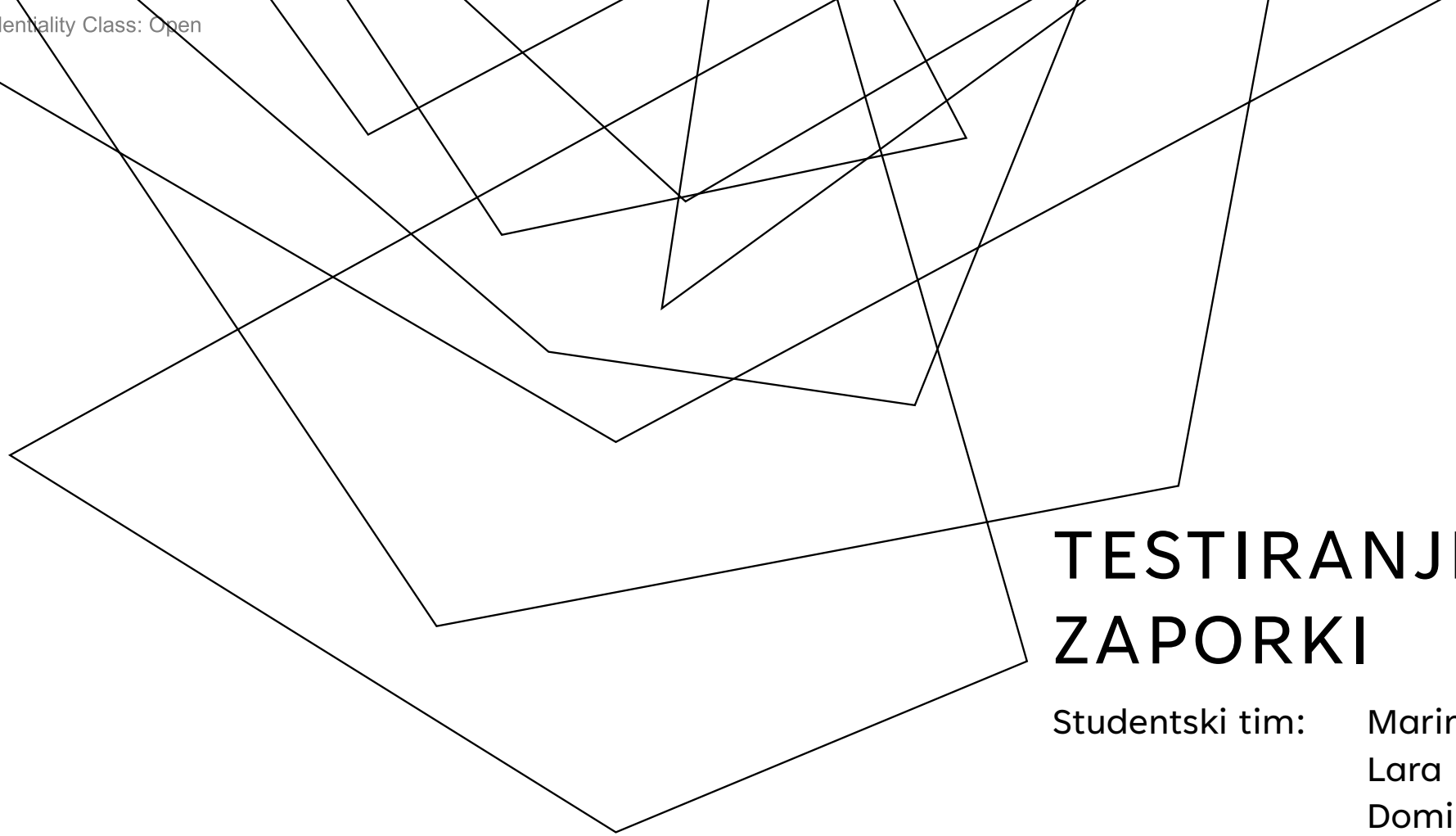

TESTIRANJE SIGURNOSTI ZAPORKI

Studentski tim: Marin Mikulčić
Lara Brečić
Dominik Topić
Sven Sonicki

Mentor: prof. dr. sc. Marin Golub

CILJ PROJEKTA

- Cilj projekta je ispitati četiri alata za oporavak zaporki kroz unaprijed definirani eksperiment
- Potrebno je osmisliti eksperiment i testirati svaki od četiri alata
- Eksperiment je potrebno provesti koristeći različite hash funkcije
- Alati koje ispitujemo:
  - **John the Ripper**
  - **Hashcat**
  - **Wfuzz**
  - **RainbowCrack**

JOHN THE RIPPER – OPIS

- John the Ripper je besplatan i otvoren izvorni alat za audite sigurnosti i obnovu zaporki
- Podržava širok spektar tipova kripto sažetaka i šifri
- Razvijen je za Unix, macOS, Windows, web aplikacije, grupne aplikacije, kao i baze podataka
- Omogućava obradu snimaka mrežnog prometa, šifriranih privatnih ključeva, datotečnih sustava, arhiva i dokumenata
- Jedan je od najčešće korištenih programa za testiranje sigurnosti zaporki

HASHCAT - OPIS

- Otvoreni alat za opravak zaporki i analizu hashev
- Poznat po izuzetno velikoj brzini rada
- Često je korišten od strane sigurnosnih istraživača, testnih timova i sistemskih administratora
- Podržava više stotina algoritama za pohranu zaporki, od zastarjelih poput md5 do modernih i složenih funkcija izvedenih iz SHA-1, SHA-2, PBKDF, poput:
  - PBKDF2-HMAC-SHA256
  - Argon2id

WFUZZ - OPIS

- Wfuzz je alat otvorenog izvora namijenjen fuzzingu web aplikacija
- Koristi u sigurnosnom testiranju s ciljem otkrivanja skrivenih resursa, funkcionalnosti i potencijalnih ranjivosti
- Radi tako da generira i šalje velik broj HTTP zahtjeva prema ciljnoj web aplikaciji
- Sustavno mijenja dijelove zahtjeva koristeći unaprijed definirane rječnike ili generirane uzorke, preko ove funkcionalnosti ćemo mi generirati zaporke

RAINBOWCRACK – OPIS

- RainbowCrack je alat za probijanje hasheva lozinki
- Baziran je na konceptu „Rainbow tablica“
- Tablica se sastoji od dugih lanaca čiji su elementi nizovi znakova
- Alat podržava više stotina algoritama za pohranu zaporki, neki od njih su:
  - LM
  - NTLM
  - MD5
  - SHA-1
  - SHA-256

HASH ALGORITMI ZA POHRANU ZAPORKI

- Za pohranu zaporki koristimo hash algoritmi
- U slučaju sigurnosnog proboja baze podataka, hashiranje sprječava izravan pristup izvornim zaporkama korisnika
- Hash funkcije uzimaju izvorne zaporke te ih irreverzibilno pretvaraju u stringove fiksne duljine koji se mogu spremati u bazu podataka
- Iz hash vrijednosti nije moguće izravno rekonstruirati izvornu zaporku

STARI HASH ALGORITMI ZA POHRANU ZAPORKI

- Algoritmi poput SHA-1, MD5 i sličnih su bili korišteni zbog:
  - Velike brzine izvođenja
  - Dobre efikasnosti na ograničenom hardveru
  - Jednostavne implementacije
  - Dovoljne razine sigurnosti za tadašnje brzine napade
- Zbog velikog rasta procesorske moći računala, korištenje ovih algoritama postaje nesigurno jer omogućuju napade grubom silom

MODERNI HASH ALGORITMI ZA POHRANU ZAPORKI

- Moderni hash algoritmi, poput SHA-256, bcrypt i Argon2id, razvijeni su s naglaskom na sigurnost
- Dizajnirani su za snažan otpor napadima grubom silom
- Sadrže podesive parametere kojima se složenost algoritama može prilagoditi performansama dostupnog hardvera
- Određeni algoritmi su dodatno memorijski zahtjevni, čime se učinkovito usporavaju i otežavaju napadi putem GPU-ova

EKSPERIMENT

- Ekperiment na kojim ćemo pokretati alate sastoji se od tri datoteke sa zaporkama:
  - Prva datoteka:
    - sadrži nasumične zaporke
    - Duljine do 8 znakova
  - Druga datoteka:
    - Sadrži 1000 najčešće korištenih zaporki
  - Treća datoteka:
    - Sadrži nasumično generirane zaporke
    - Duljine od 16 do 20 znakova.
- Eksperiment se provodi korištenjem više različitih hash algoritama

BRZINE ALATA

Benchmark alata

- Prije pokretanja eksperimenta su alati ispitani kako bi se saznala njihova teoretska brzina
- Dobiveni rezultati su stavljeni u tablicu

Dobivene brzine

Brzina u hash/sek	John the ripper	Hashcat	RainbowCrack	Wfuzz
Nešifrirani tekst	10 450 000	-	-	576
SHA-1	11 246 000	39 564 100 000	~1000	376
SHA-1 + salt	19 952 000	39 363 500 000	-	-
SHA-256	11 494 000	14 720 700 000	-	355
MD5	21 760 000	109 200 000 000	~700	330
PBKDF2-HMAC-SHA256 (600000)	25	9 279	-	-
Argon2id (m=131072, t=3, p=1)	15	458	-	-

JOHN THE RIPPER – REZULTATI EKSPERIMENTA

	SHA1	Salted-SHA1	Argon2i	PBKDF2-HMAC-SHA256
bad (2400)	1329	17	-	-
common (1000)	1000	944	405	556
locker (1000)	-	-	-	-

*Broj predstavlja količinu probijenih zaporki unutar svake od tri datoteka za svaki od hash algoritama

HASHCAT – REZULTATI EKSPERIMENTA

Hashing algoritmi	bad	common	locker
SHA-1	1847	985	-
SHA-1 + salt	667	986	-
PBKDF2-HMAC-SHA256	0	703	-
Argon2id	0	979	-

*Broj predstavlja količinu probijenih zaporki unutar svake od tri datoteka za svaki od hash algoritama

WFUZZ – REZULTATI EKSPERIMENTA

Server side hashing	Nehashirano	MD5	Argon2id
Common(901)	269 (3874s)	208 (2363s)	70 (2142s)

*Broj predstavlja količinu probijenih zaporki unutar svake od tri datoteka za svaki od hash algoritama

RAINBOWCRACK – REZULTATI EKSPERIMENTA

Datoteka	bad	common	locker
SHA-1	1224 (12.5 h)	964 (2.2 h)	0

*Broj predstavlja količinu probijenih zaporki unutar svake od tri datoteka za svaki od hash algoritama

ZAKLJUČAK

Zaporke

- Preporučljivo je odabrati zaporke dulje od 8 znakova uz obavezno korištenje brojeva, velikih slova I specijalnih znakova
- Potrebno je izbjegavati zadržavanje automatski dodijeljenih zaporki te ih zamijeniti vlastitima
- Pri kreiranju zaporke važno je ne koristiti uobičajne riječi ili lako predvidibe pojmove

Hash algoritmi

- Prilikom odabira hash algoritma, ključno je koristit modern standard koji su dizajnirani da budu vremenski I memorijski zahtjevni
- Za ispravnu implementaciju algoritma nužno je pravilno podesiti konfiguracijske parametre te koristiti jedinstvanu vrijednost sol (engl. *salt*)