DEFACK

GSE GUIDE SHARE EUROPE UK REGION

Mainframe passwords revisited: impact of new security mechanisms

Costin Enache, Chad Rikansrud, Nigel Pentland

GSE UK Security Working Group – 4th February 2021

This slide deck: ~20 minutes
Panel discussion and Q&A: ~40 minutes

Who are we?

- Chad Rikansrud

IT security consultant, Director of North American Consulting Services for BMC, performing mainframe security assessments, exploit development, and penetration tests for some of the world's largest organizations for 20+ years. chad_rikansrud@bmc.com

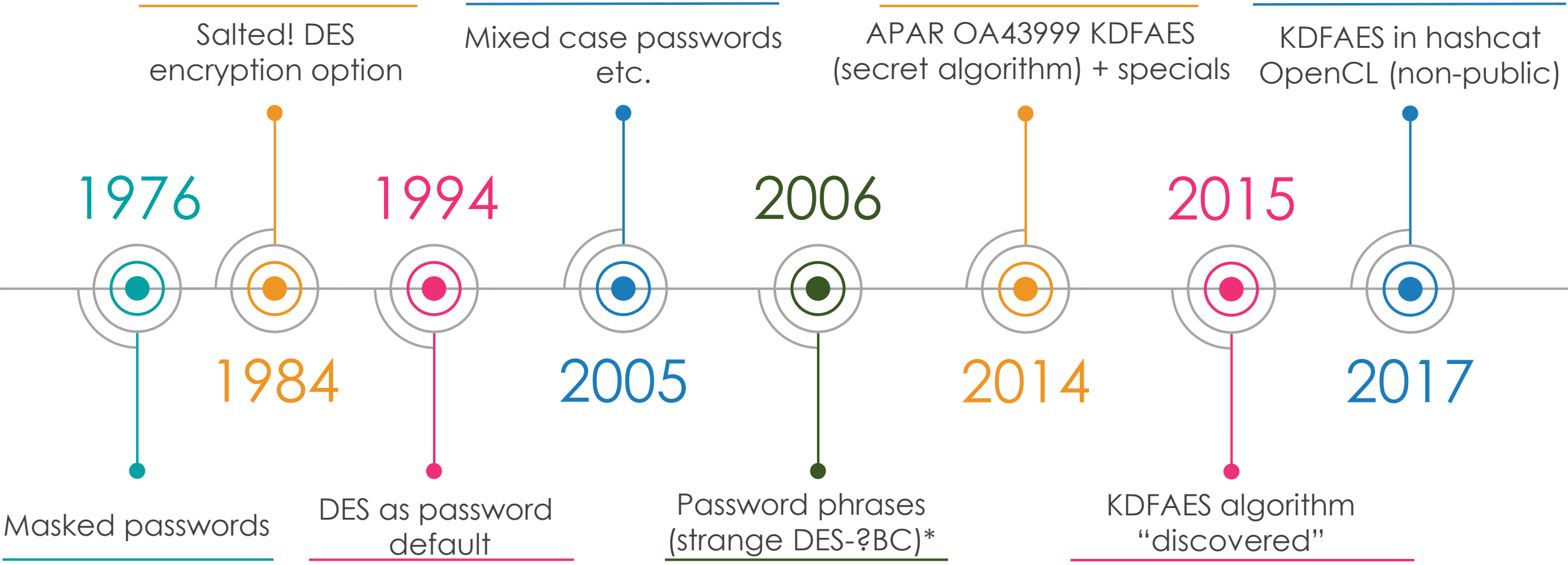
- Nigel Pentland

Senior Security Analyst at NAB, retired. The authority on mainframe passwords for many years and the author of the original analytics tools that saved many of us. nigel@nigelpentland.net

- Costin Enache

IT security consultant and occasional developer, MD at Detack.de, been working with mainframe security for 20+ years; author of EPAS, a toolset for password analytics that includes RACF with KDFAES support. costin@detack.de

History



*IBMUSER:AGoodPassword2001 = AGoodPassword2002 = 1D80A5E7A1C14709DD0C44377CBD8303E0

The KDFAES Algorithm

- **Step #1:** Calculate input for key derivation, Ki

  Passwords: Ki[8] = Old DES password = DES(K=password, In=username); permits **one-way migration** of old DES

  Password phrases: Ki[40] = SHA256(phrase) + length(phrase); no migration possible for DES phrases

- **Step #2:** Key derivation based on custom IBM version of PBKDF2-SHA256; 45008 rounds*

  Kd = IBM-PBKDF2-SHA256(RANDOM_SALT[16],Ki,45008)

- **Step #3:** Encrypt, by using AES256, the user name with the derived key

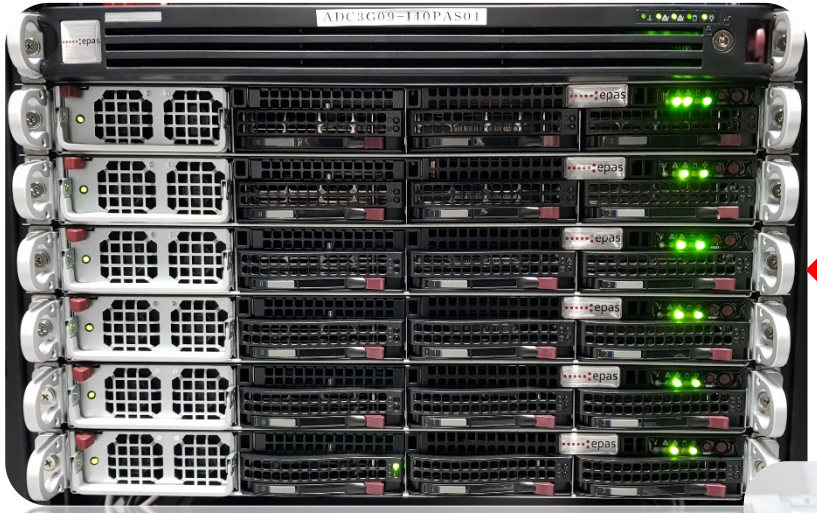
  Hash[32] = AES256(K=Kd, In=username) + RANDOM_SALT[16]

- **The icing on the cake:** It is quantum-safe ☺ Both SHA256 and AES256 are believed to be safe.

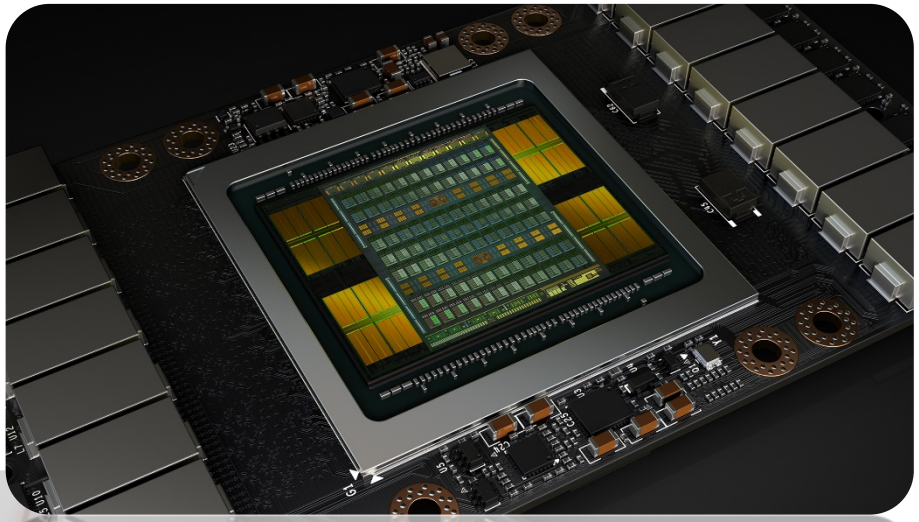
*The algorithm includes 2 parameters (**memory and repetition factors**) which can be customized and get stored within the hash. This allows the key derivation step to be future-proof, i.e., get slower. Not in use for now, but if maxed out, the computation time is increased ~4500 times

Link: https://github.com/openwall/john/blob/bleeding-jumbo/src/racf_kdfaes_fmt_plug.c

The Tools: Cluster, Big Server, Amazon EC2



Examples based on 1xGPU

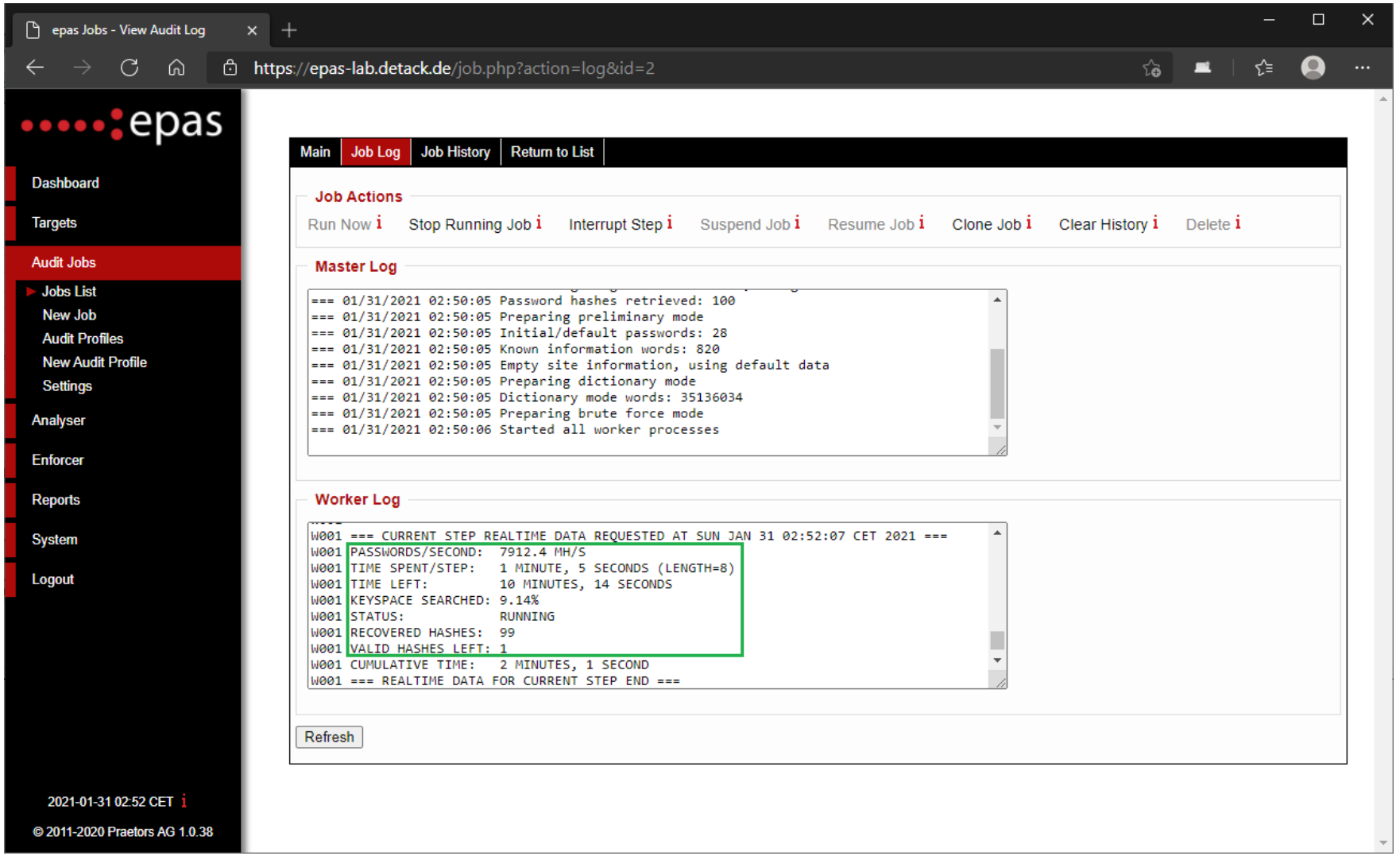


- 6 x GPU Worker Cluster

- 10 x GPU Worker, Single Unit

- 1 x Amazon EC2 p3.16xlarge

DES Performance: Brute Force



DES: Brute Force Probing

Upper Case

Standard Character Set

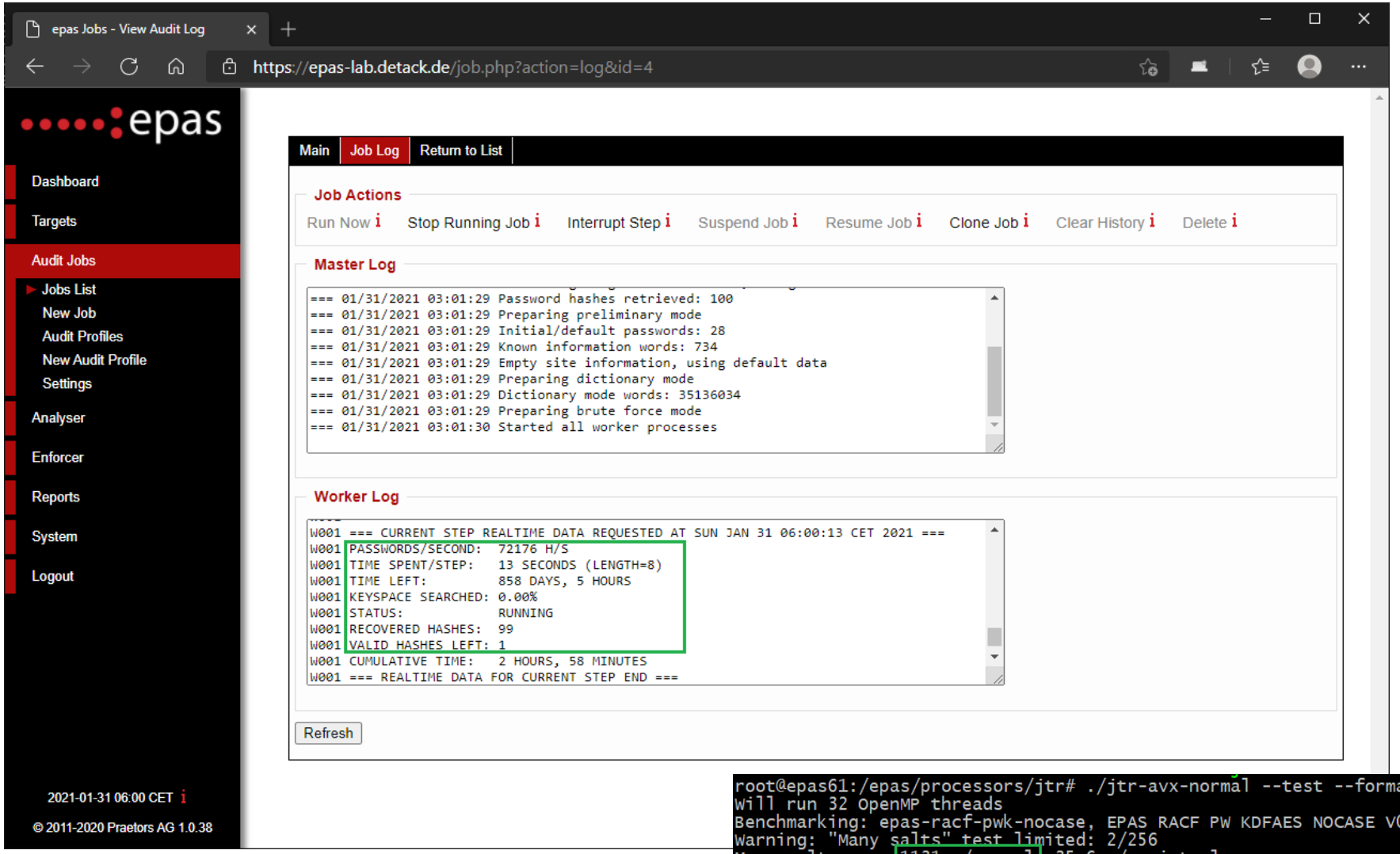
8-Character Password

Users: 1

Guaranteed Recovery: 13 min

Worker: 1 x Nvidia 3090 GPU

KDFAES Performance: Brute Force



KDFDES: Brute Force Probing

Upper Case

Standard Character Set

8-Character Password

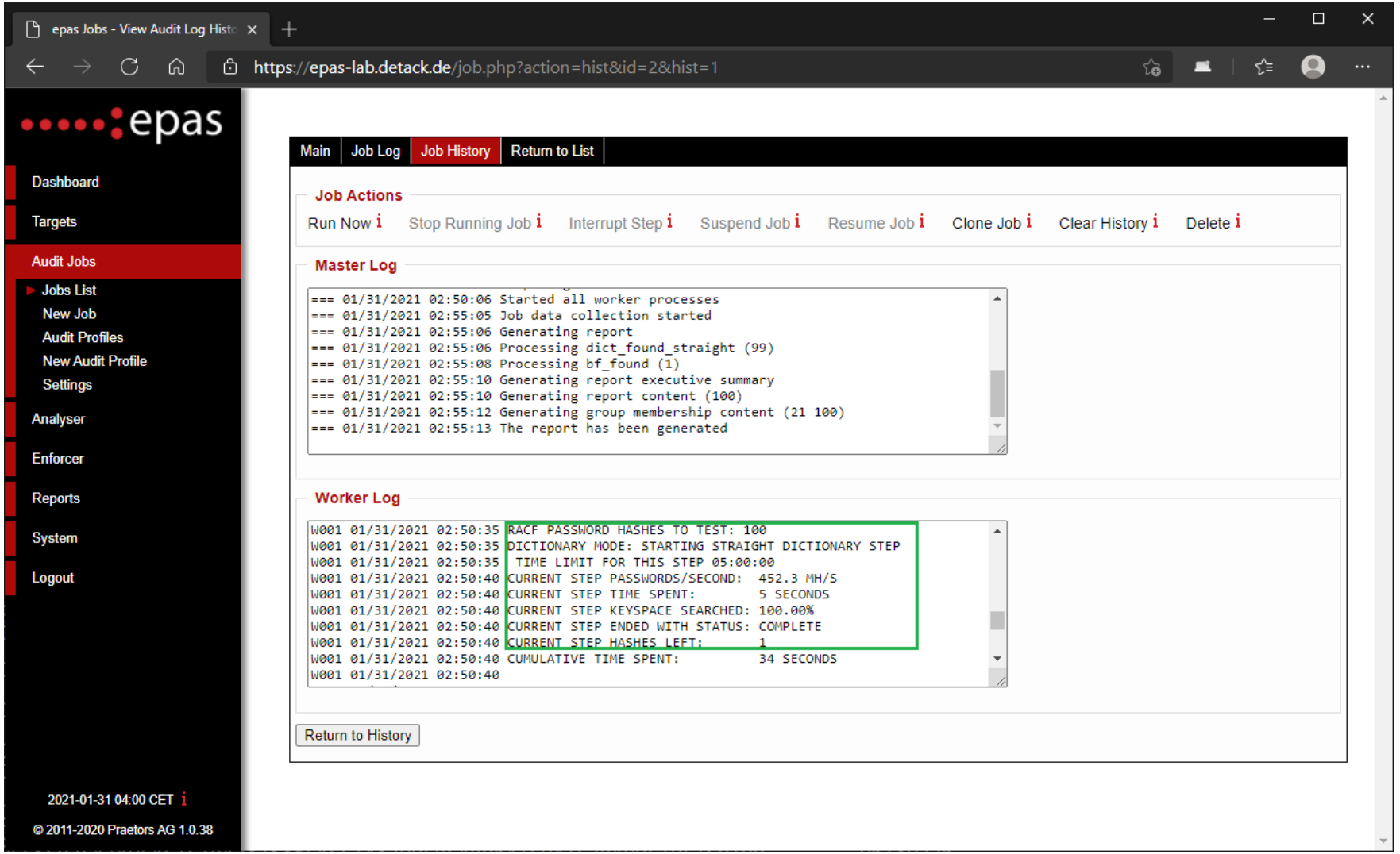
Users: 1

Guaranteed Recovery: 858 days

Worker: 1 x Nvidia 3090 GPU

```
root@epas61:/epas/processors/jtr# ./jtr-avx-normal --test --format=epas-racf-pwk-nocase
Will run 32 OpenMP threads
Benchmarking: epas-racf-pwk-nocase, EPAS RACF PW KDFAES NOCASE V0.1 [DES-PBKDF2-SHA256-AES 128/128 AVX 4x]... (32xOMP)
Warning: "Many salts" test limited: 2/256
Many salts:     1131 c/s real,  35.6 c/s virtual
Only one salt:  1119 c/s real,  35.6 c/s virtual
```

DES Performance: Wordlist



DES: Wordlist Probing

Upper Case

Standard Character Set

8-Character Password

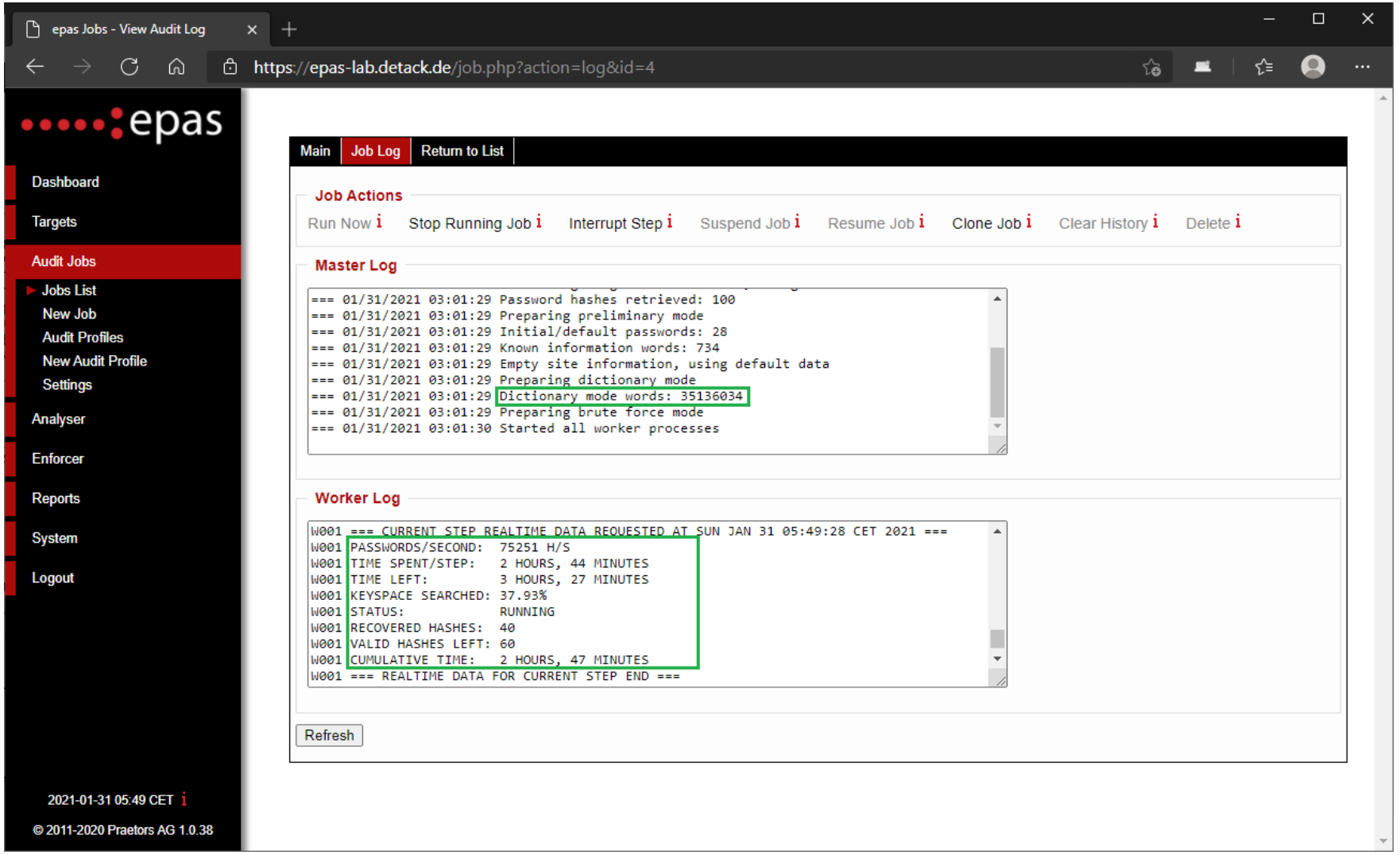
Users: 100

Dictionary: 35,136,034

Complete Keyspace: 5 seconds

Worker: 1 x Nvidia 3090 GPU

KDFAES Performance: Wordlist



KDFDES: Wordlist Probing

Upper Case

Standard Character Set

8-Character Password

Users: 100

Dictionary: 35,136,034

Complete Keyspace: 6 hours

Worker: 1 x Nvidia 3090 GPU

Numbers Compared

- **3090 GPU Worker Instance (HW Price: €30k)**

	1 x GPU		10 x GPU	
RACF KDFAES	80,449	H/s	800,210	H/s
RACF DES	7,467	MH/s	73,716	MH/s
NTLM	116,602	MH/s	1,149,035	MH/s

- **Amazon EC2 p3.16xlarge (HW Price: €20/h)**

	1 x GPU		8 x GPU	
RACF KDFAES	71,193	H/s	569,550	H/s
RACF DES	5,809	MH/s	46,564	MH/s
NTLM	102,202	MH/s	818,215	MH/s

- **Apparently: Average of 85,000 times harder to crack KDFAES than DES via brute force attacks**
- **And 1,500,000 times harder than NTLM**

2010 DES

- **8 Core server system, CPU-only ~ €3000**

  Off the shelf JtR:
  8 x 778,752 = 6,230,016 H/s
  Smarter JtR (EPAS: Bitslice DES, AVX/SSE2):
  8 x 24,925,141=199,401,128 H/s ~aprox. 33x faster

2020 KDFAES

- **Server with 3090 GPU ~ €3000**

  OpenCL / CUDA:
  80,449 H/s

- **Amazon 8xGPU p3.16xlarge for ~ 1 week ~€3000**

  8 x 71,193 = 569,550 H/s

- **Between 11 and 2,500 times harder to crack a RACF password in 2020 (KDFAES) compared to 2010 (DES)**

Are RACF password hashes now safe?

- Brute-force attacks are no longer a viable option
- Current OpenCL hardware and cloud resources still provide a good speed of between 80,000 and 800,000 H/s
- Hash cracking is still possible, but the keyspace must be trimmed; attackers were doing this anyway
- Anatomy of a real attack:

  Minimize the number of users: target service, privileged, developer, TSO accounts only

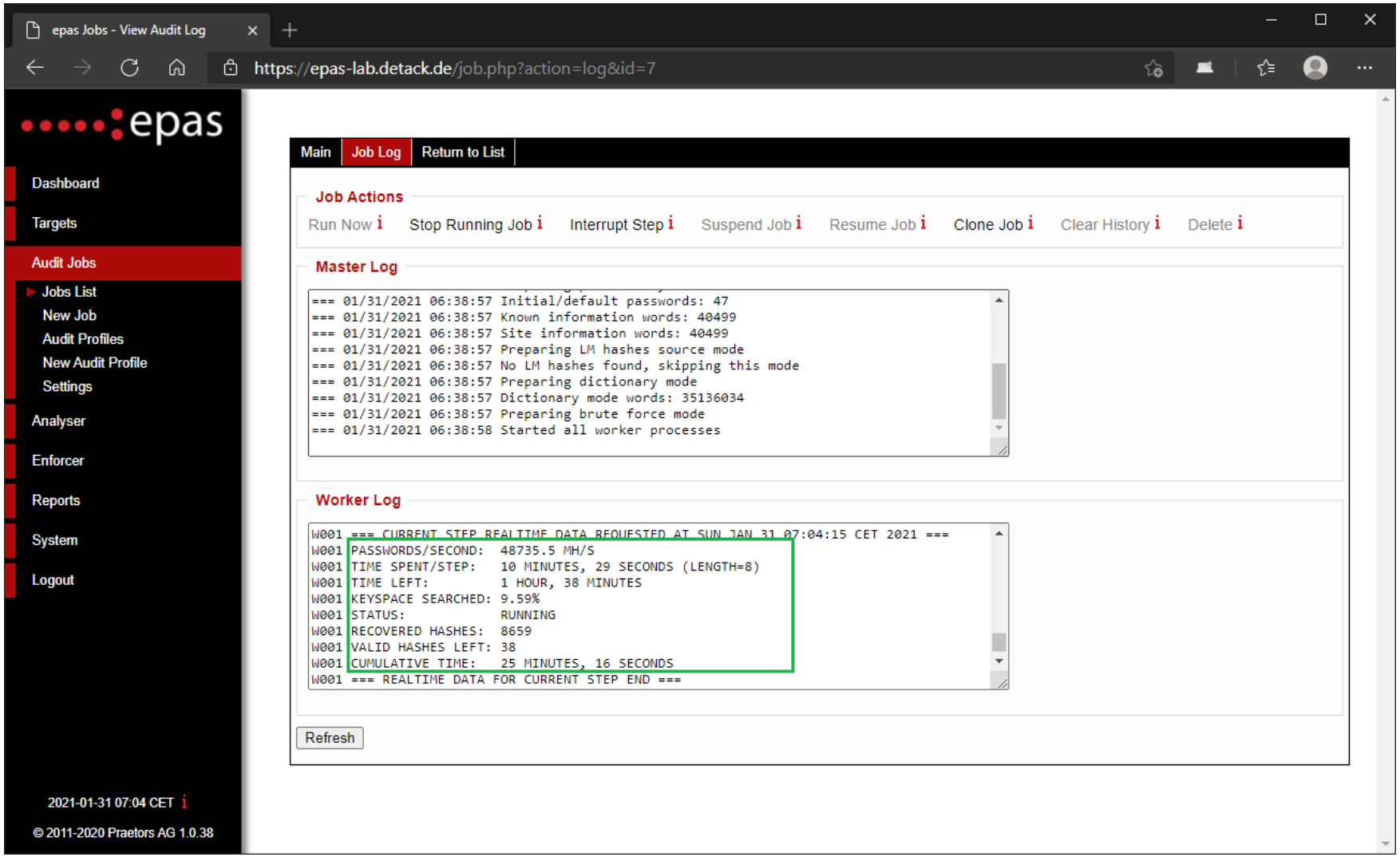
  Use wordlists customized for the current target: leaked passwords, company branding

- Event better, what we do when we run RACF security assessments:

  Target a weaker system first, say Active Directory (unsalted), and use the recovered passwords as wordlist

  Many users will have identical or similar passwords; nobody needs to crack all of them, one is usually enough

NTLM: Fast Cracking, No Salt



NTLM: Full Audit

Mixed Case

Standard Character Set

12-Character Password

Users: 10,000

Performance: ~80% in 2 hours

Worker: 1 x Nvidia 3090 GPU

NTLM: Results

NTLM: Full Audit

Mixed Case

Standard Character Set

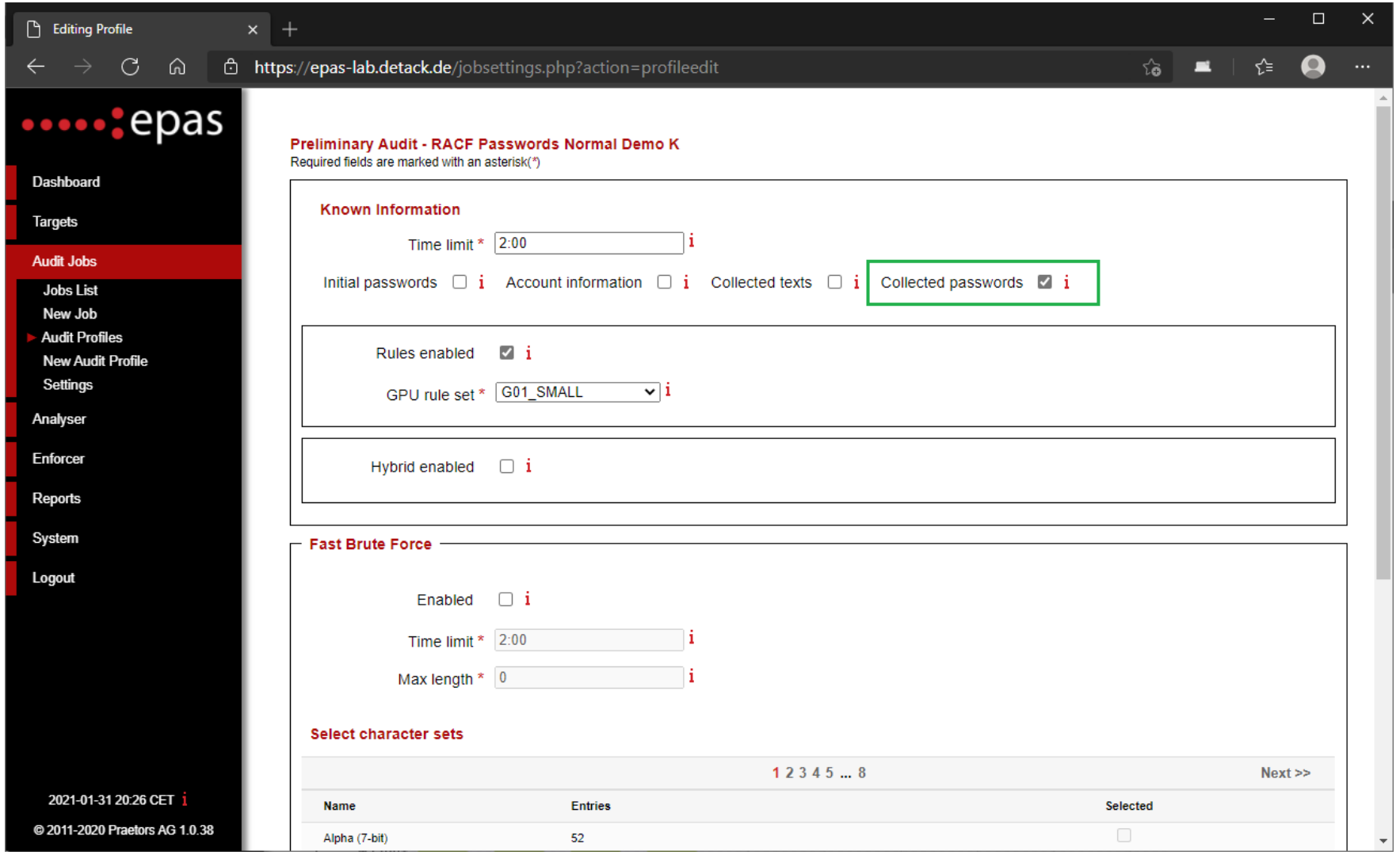
12-Character Password

Users: 10,000

Performance: ~80% in 2 hours

Worker: 1 x Nvidia 3090 GPU

KDFAES: Use Results from NTLM as Wordlist



KDFAES: Preliminary Audit

Upper Case

Standard Character Set

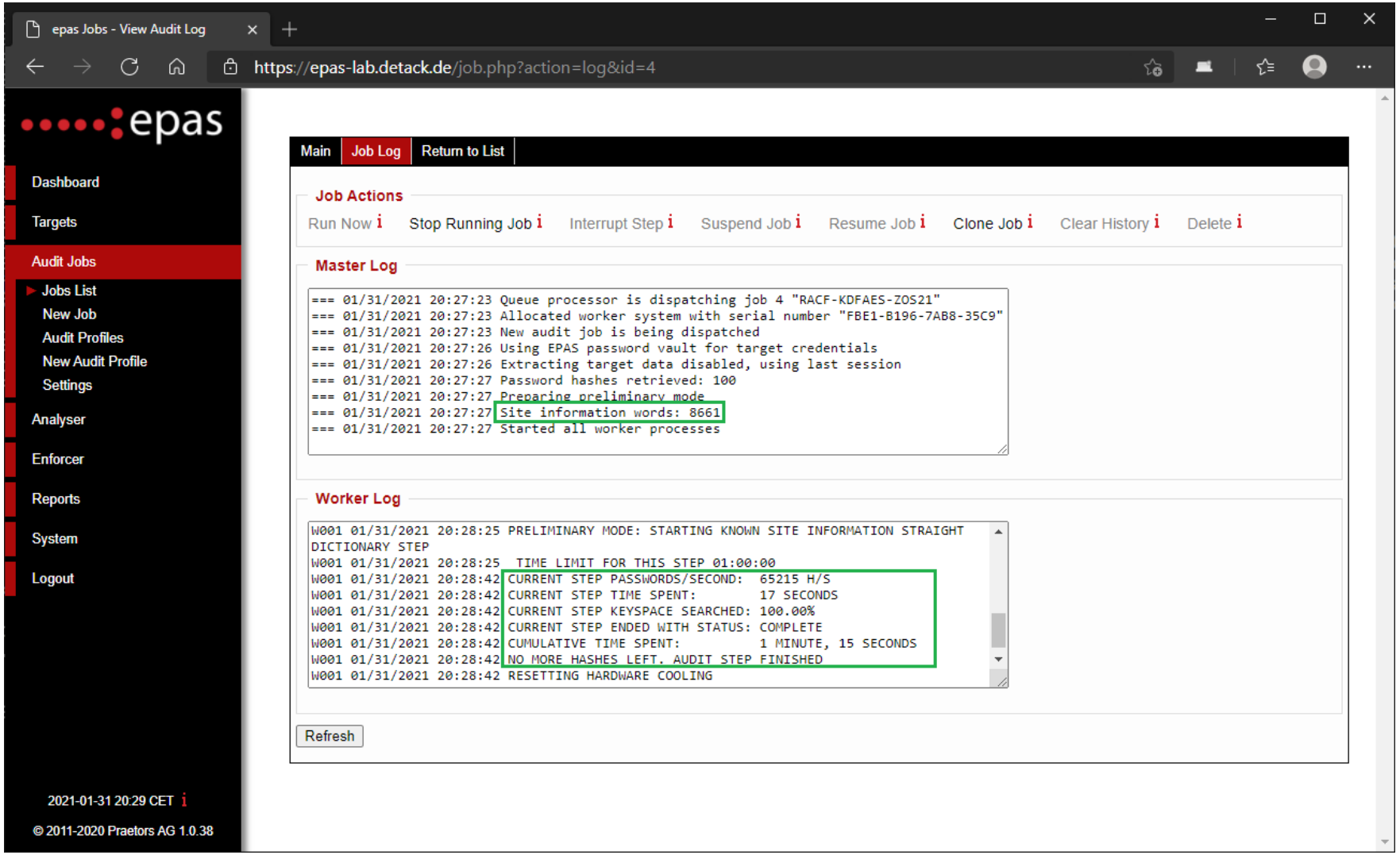
8-Character Password

Users: 100

Dictionary: 8,661

Worker: 1 x Nvidia 3090 GPU

KDFAES: Results



KDFAES: Preliminary Audit

Upper Case

Standard Character Set

8-Character Password

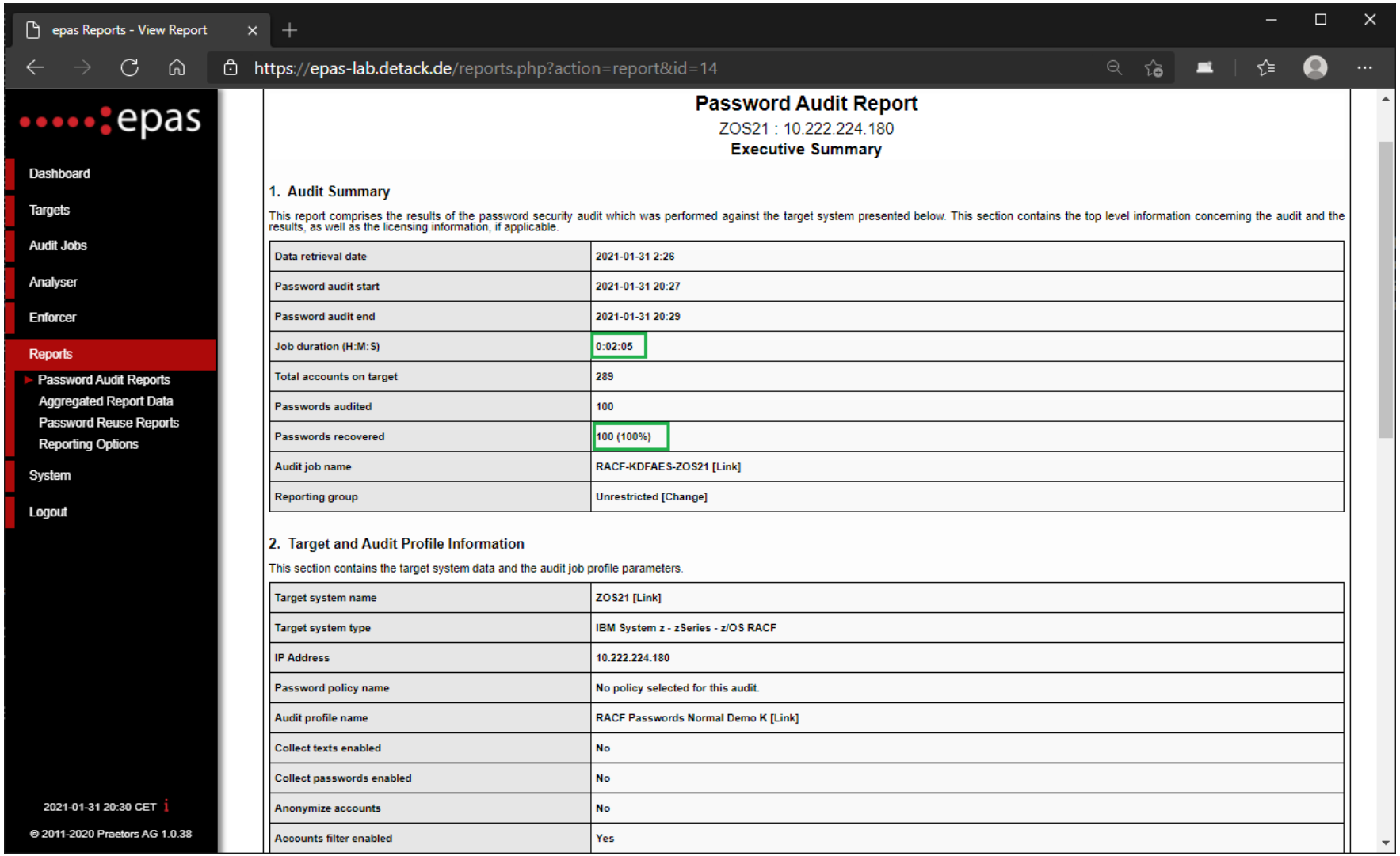
Users: 100

Dictionary: 8,661

Performance: 100% in 2 min

Worker: 1 x Nvidia 3090 GPU

KDFAES: Results



KDFAES: Preliminary Audit

Upper Case

Standard Character Set

8-Character Password

Users: 100

Dictionary: 8,661

Performance: 100% in 2 min

Worker: 1 x Nvidia 3090 GPU

In Brief

- Attacks against password hashes on RACF are still possible, but harder and yield less cracked passwords
- As long as weaker systems exist in the environment, “a chain is only as strong as its weakest link”
- Mainframe security efforts no longer make sense isolated from the rest
- Password cracking is usually NOT the way mainframes are hacked, but it is in 90% of the cases instrumental, so make sure strong passwords are used, and are not the same or similar to other, weaker systems
- Password spraying can be used to hack mainframes too, so make sure that leaked passwords are detected
- N.B. RRSF: Make sure all systems use KDFAES, as the password is transmitted in clear text and hashed by each RACF instance separately with KDFAES or DES, as configured.

Questions we ask ourselves …

1. Have you encountered any real life cases where password cracking of exposed RACF db was the reason, or at least instrumental in a successful mainframe security attack (no names)? How about employing such methods yourself as an auditor / penetration tester?
2. Password spraying, and leaked passwords in general: what is the actual danger, and what is the awareness we have today in enterprises, in general, but also specifically for mainframe users? Would alerting prevent such an attack?
3. Stronger password hashing: Are we better off now? Hash cracking is only a matter of interest if the RACF database is exposed. Does it help security, by making the system safer, does it help people to use weaker passwords undetected by analytics tools?
4. Adding another factor besides the password would make things more secure, at least for interactive users. Is IBM MFA AZF????1 getting any traction? Acceptance is bad amongst users, PSD2 has been challenged in Germany, do we know of any adaptive/conditional authentication for RACF?
5. Obscurity vs. security, would it make sense that intended changes in password crypto is based on standards and fully disclosed, instead of keeping it secret? Would eliminating potential mistakes – see the DES password phrases – outweigh the potential / imaginary risks or disclosing the algo?