PathWell: Password Topology Histogram Wear-Leveling

June 2014
BSides Asheville

Hank Leininger – KoreLogic

https://www.korelogic.com/

tl;dr: Make users' passwords 5-6 orders of magnitude harder to crack.

Agenda

My Background

Classic Password Cracking

Classic Defenses

Recent Trends

PathWell

Examples

Next Steps

My Background

Hank Leininger <hlein@korelogic.com>
D24D 2C2A F3AC B9AE CD03 B506 2D57 32E1 686B 6DB3

Played defense as a sysadmin / security admin since the mid 90's.

Have been doing security consulting since 2000; co-founded KoreLogic in 2004.

We created the Crack Me If You Can contest at DEFCON; 2013 was its 4th year running.

I also run the MARC mailing list archive site: http://marc.info/

KoreLogic SECURITY

PathWell Background

- I had the ideas for the following analysis, and the enforcement approach described later, in late 2010 or so.

- In 2013 won a DARPA Cyber FastTrack contract to flesh out the research, design, and build a proof of concept.

- My coworkers did most of the actual work developing the PathWell PoC.

Agenda

My Background

Classic Password Cracking

Classic Defenses

Recent Trends

PathWell

Examples

Next Steps

KoreLogic SECURITY

Classic Password Cracking

Offline cracking:

- Naive bruteforce (impractical)
- Wordlists
- Mangling rules

Popular classic tools: Crack, L0phtCrack, John the Ripper

Agenda

My Background

Classic Password Cracking

Classic Defenses

Recent Trends

PathWell

Examples

Next Steps

Classic Defenses

- Password complexity rules
  - Minimum length
  - Character classes
- Password rotation
  - History retention
- Better hash types (rarely implemented)

Agenda

- My Background
- Classic Password Cracking
- Classic Defenses
- **Recent Trends**
- PathWell
- Examples
- Next Steps

Recent Trends: Attacker Advantage

Today the deck is stacked in the attackers' favor.

- Enterprise software vendors haven't moved to stronger hash types.
- Moore's law has helped attackers tremendously.
- Existing defenses (password policies) have lead to exploitable predictability.
- Systems with design flaws are vulnerable to pass-the-hash attacks, which can make password cracking unnecessary.

Recent Trends: Attacker Advantage

- **Legacy systems** mean we are still using hash types we have known were too weak for many years now.
  - **UNIX DES** was replaced with better things in free UNIXes since the 90's, but it's only fairly recently that commercial UNIXes have gotten better options.
  - **NTLM**, the strongest hash type offered by the latest Microsoft products, was too weak to use even when it was new in 1993.
  - **{SSHA}**, single-round salted SHA-1, is the best offered by many enterprise LDAP directories.
- **GPU power has made selective brute-forcing practical** for these weak hashes, even for quite long password lengths.

Recent Trends: Attacker Advantage

Password Policies create new exploitable predictability:

- **Complexity rules** result in users choosing and placing their uppers, lowers, numbers, and specials in predictable ways:
  - Capitalize the first letter(s) of words (**WeakSauce**)
  - Numbers likely to be at the end, and to be a year (**WeakSauce2014**)
  - Add specials to the end (**WeakSauce2014!**)
  - Predictable character choice - '**!**' is the most common special character by a huge margin

- **Password rotation** results in users simply modifying their old passwords in predictable ways:
  - “**Oct0b3r!**” → “**N0v3mb3r!**”
  - “**Winter2013!**” → “**Spring2014!**”
  - “**qWErt78()**” → “**wERty89)_**”

KoreLogic SECURITY

Naive Brute Force

For about $2,000 you could build a machine with three AMD 7970 GPUs.

Each GPU can try ~6,746,000,000 candidate plaintexts per second against a list of NTLM hashes, which means about 20 billion per second for the system.

That machine could try all possible 8-character NTLM passwords using printable ASCII (95^8) in 3.8 days.

But as you add length, the time gets longer quickly:

- 9 characters: 360 days (or 18 days for 20 machines)
- 10 characters: 94 years
- 11 characters: 8,900+ years
- 15 characters: 734 billion years

Selective Brute Force – Password Patterns

- Rather than testing all possible passwords, pick some specific subsets, or patterns, and try all passwords that fit that pattern (“topology”).
- For instance, "P4ssword13!", "N0vember24@", "R3dskins99#" all use the same pattern: Uppercase, number, 6 lowercase, 2 numbers, special.
- We will use the same notation as the Hashcat tools:
  - 'u' to represent "any uppercase letter"
  - 'l' for "lowercase letter"
  - 'd' for "digit"
  - 's' for "special" (punctuation)
- The above example is then “?u?d?l?l?l?l?l?d?d?s”, or just “udlllllldds” for short.

Selective Brute Force – Password Patterns

- u, l, d, s = four possible character sets per password character.
- 8 character password: 4^8, or 65,536 possible topologies
- 9 character: 4^9 = 262,144
- 10 character: 4^10 = 1,048,576
- 11 character: 4^11 = 4,194,304
- The 11-character topology udllllllldds has 265 trillion possible passwords (A0aaaaaa00! - Z9zzzzzz99~).

- Our example cracking machine, which would take 8,900 years to exhaust the entire 11-character space, could bruteforce that one topology in just **3.6 hours**.

Predictable Password Topologies

- The question then is: do users bias towards certain common password topologies?
- If you can guess which patterns users have over-used, you can effectively bruteforce just those topologies, and crack a disproportionate number of passwords.
  - In reality you would likely combine that with wordlists, mangling rules, and character frequencies to further optimize your attack.
- We analyzed the passwords we had cracked from several different enterprise assessments, looking for frequently used topologies.

Predictable Password Topologies

- The question then is: do users bias towards certain common password topologies? **[Spoiler: OMG YES THEY DO.]**
- If you can guess which patterns users have over-used, you can effectively bruteforce just those topologies, and crack a disproportionate number of passwords.
  - In reality you would likely combine that with wordlists, mangling rules, and character frequencies to further optimize your attack.
- We analyzed the passwords we had cracked from several different enterprise assessments, looking for frequently used topologies.

Sample Organization #1: Fortune 100 Company

- 263,356 of 263,888 NTLM logins cracked (including histories) – over 99%
- 7,308 unique topologies found

Sample Organization #1: Fortune 100 Company

- 263,356 of 263,888 NTLM logins cracked (including histories) – over 99%
- 7,308 unique topologies found
- Most popular topologies:
  - 33,458 ullllldd (8 character)
  - 33,394 ulllllldd (9 character)
  - 27,898 ullldddd
  - 19,190 ullllllldd
  - 13,204 ulllldddd

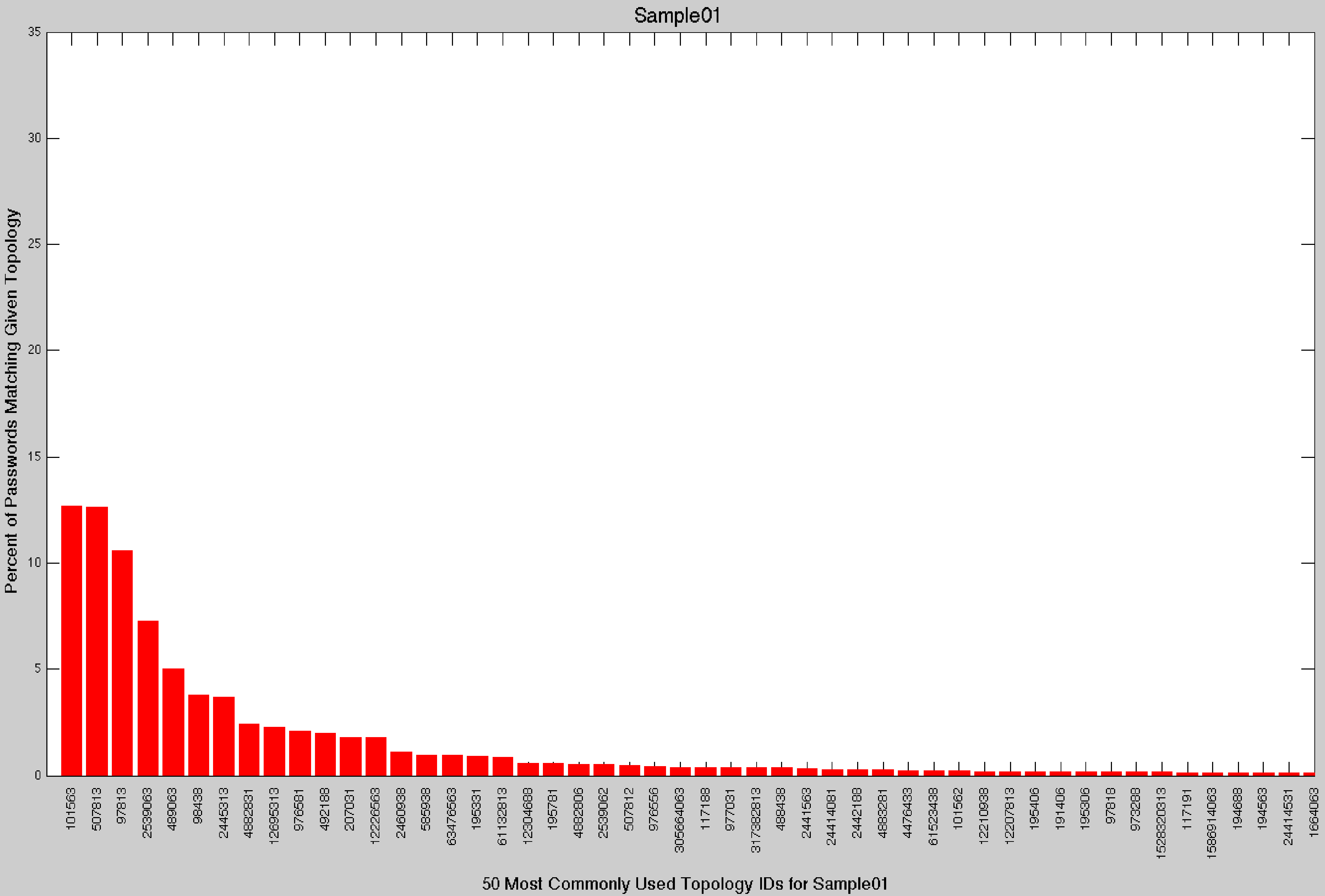
Sample Organization #1: Fortune 100 Company

- 263,356 of 263,888 NTLM logins cracked (including histories) – over 99%
- 7,308 unique topologies found
- Most popular topologies:
  - 33,458 ullllldd (8 character) – 12.7%
  - 33,394 ulllllldd (9 character) – 12.7%
  - 27,898 ullldddd – 10.6%
  - 19,190 ullllllldd – 7.3%
  - 13,204 ulllldddd – 5.0%

Sample Organization #1: Fortune 100 Company

- 263,356 of 263,888 NTLM logins cracked (including histories) – over 99%
- 7,308 unique topologies found
- Most popular topologies:
  - 33,458 ullllldd (8 character) – 12.7%
  - 33,394 ulllllldd (9 character) – 12.7%
  - 27,898 ulllddddd – 10.6%
  - 19,190 ullllllldd – 7.3%
  - 13,204 ullllddddd – 5.0%
- The top 5 patterns are used by a total of 48% of all users.
- The top 100 patterns are used by a total of 85% of all users.
- 99.9% of passwords meet their complexity requirements
  - They had recently increased their min length to 9.
  - Some history entries still had 8-char passwords.
  - Look at how similar the top 8-char topologies are to the top 9-char ones! They just added one lowercase letter (used a longer word).

Sample Organization #1:



Sample Organization #2: Fortune 500 Company

- 419,287 of 449,192 NTLM logins cracked (including histories) – 93%
- 14,266 unique topologies found

Sample Organization #2: Fortune 500 Company

- 419,287 of 449,192 NTLM logins cracked (including histories) – 93%
- 14,266 unique topologies found
- Most popular topologies:
  - 19,200 ullllldd (8 character)
  - 17,914 ullllldds (9 character)
  - 14,025 ulldddds
  - 12,477 ullllllds
  - 9,216 ullsdddd

KoreLogic SECURITY

Sample Organization #2: Fortune 500 Company

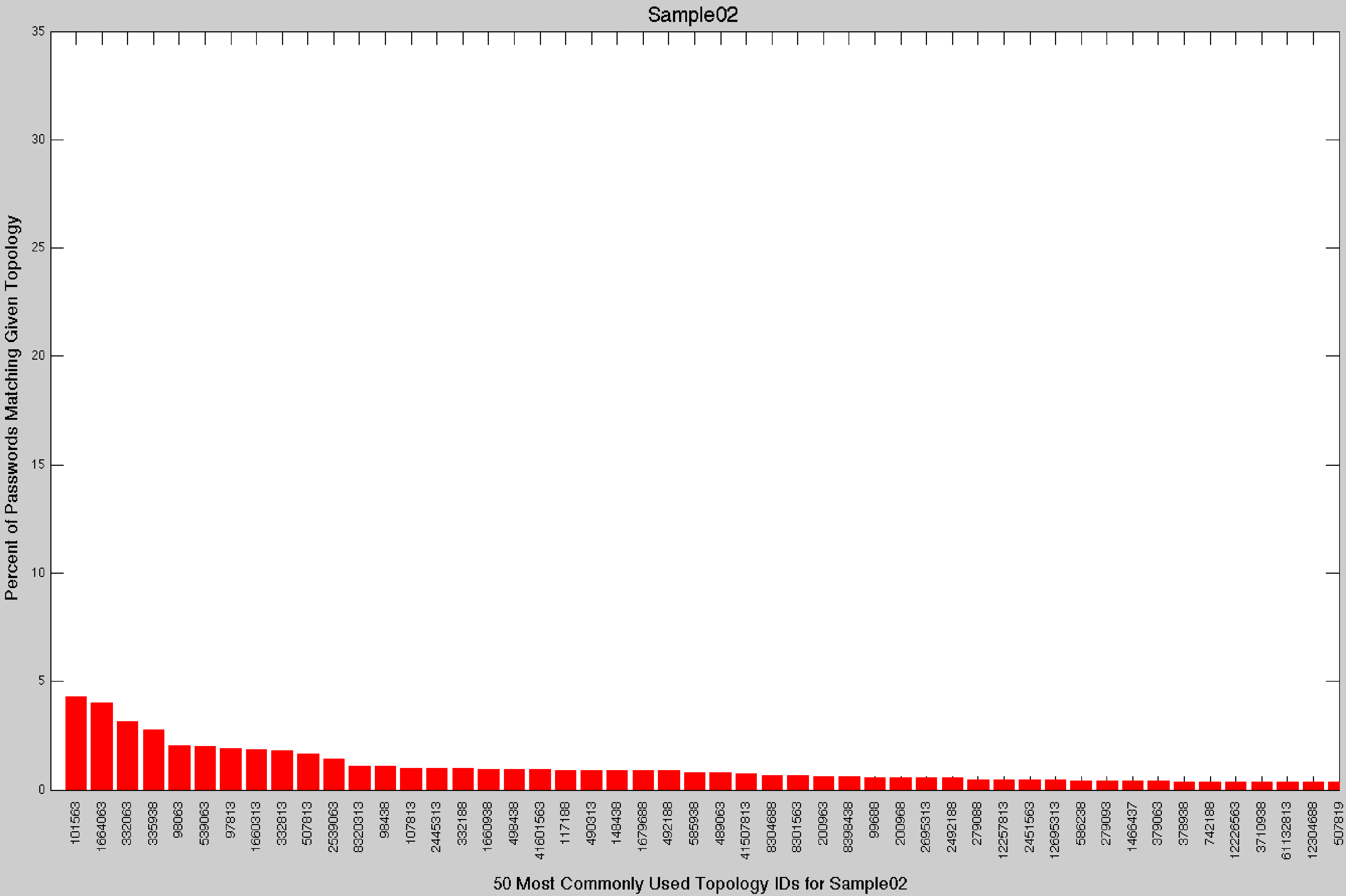
- 419,287 of 449,192 NTLM logins cracked (including histories) – 93%
- 14,266 unique topologies found
- Most popular topologies:
  - 19,200 ullllldd (8 character) – 4.3%
  - 17,914 ullllldds (9 character) – 4.0%
  - 14,025 ulldddds – 3.1%
  - 12,477 ullllllds – 2.8%
  - 9,216 ullsdddd – 2.1%

Sample Organization #2: Fortune 500 Company

- 419,287 of 449,192 NTLM logins cracked (including histories) – 93%
- 14,266 unique topologies found
- Most popular topologies:
  - 19,200 ullllldd (8 character) – 4.3%
  - 17,914 ullllldd**s** (9 character) – 4.0%
  - 14,025 ullddds – 3.1%
  - 12,477 ulllllld**s** – 2.8%
  - 9,216 ullsdddd – 2.1%
- Top 5 topologies crack 16% of all passwords.
- The top 100 topologies are used by a total of 62% of all users.
- They too had recently strengthened their requirements – longer minimum and required a special.

KoreLogic SECURITY

Sample Organization #2:

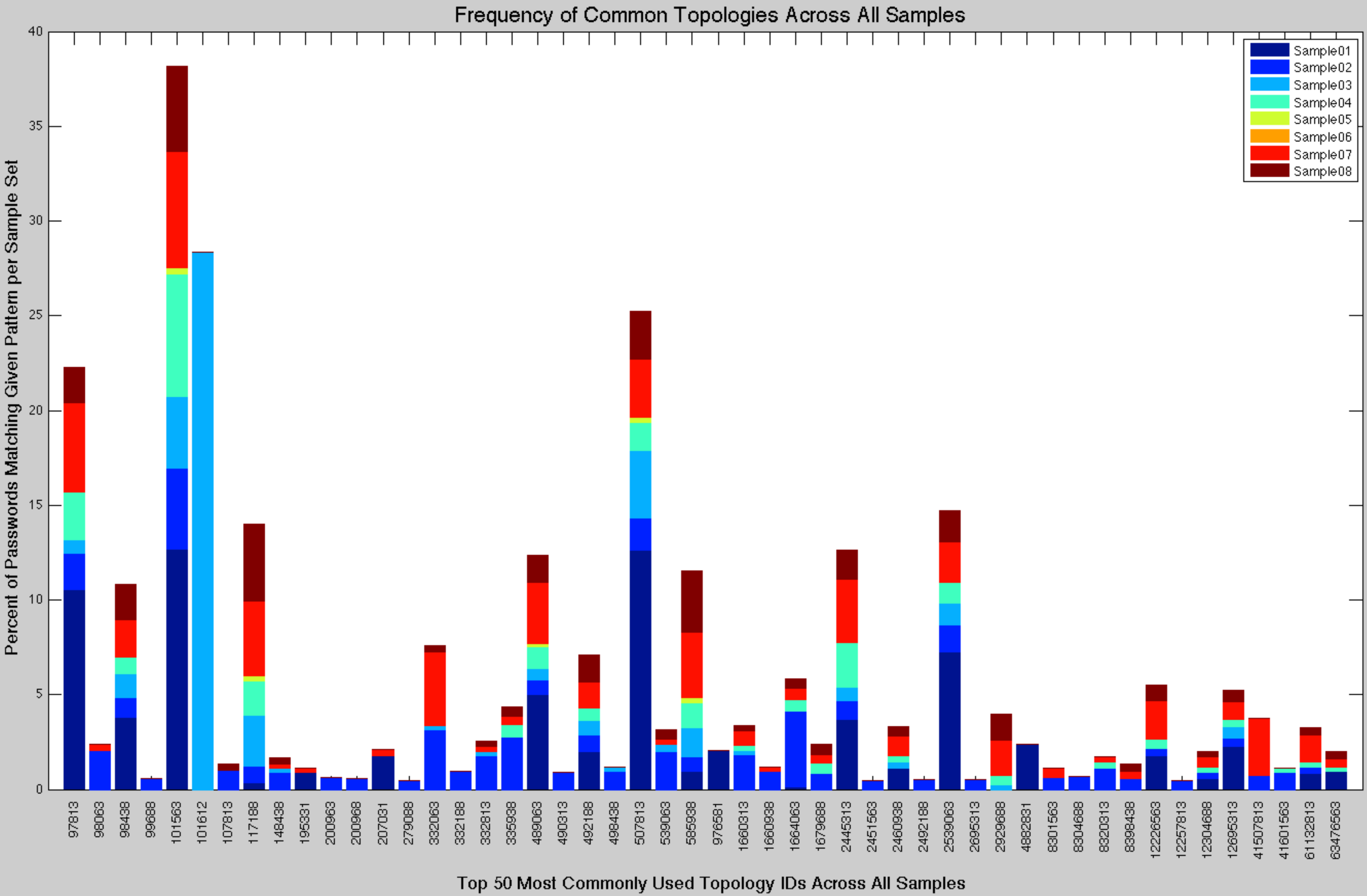


Similarities Across Organizations

We analyzed the password topologies used in 8 different enterprises of 4,000 or more logins where we had cracked more than 90% of all password hashes.

We found that they had many popular topologies in common.

Similarities Across Organizations



Things the Data Told Us

This data confirmed things we had long observed anecdotally:

- Users will pick the lowest-common-denominator that will be allowed by policies.
- When required to use 3 of 4 character classes, the most popular is: one upper, then several lowers, then 2-4 digits.
- If required to use 4 of 4 charsets, users just add a special to the end. (And most often that special character is '!')
- If the minimum length increases, users are most likely to just use a longer base word, adding a lowercase letter.
- User behavior trends apply across organizations.

Bottom line: Complexity rules don't help as much as enterprises think they do.

How about harder passwords?

How about 15-character passwords with minimum 2 uppercase, 2 lowercase, 2 digits, 2 specials?

- To brute-force the entire keyspace would take hundreds of trillions of years. It is tempting to think that they “can't be cracked”.
- But what if the attacker targets popular topologies?
  - I would guess one of the top-5 patterns would be ullllsullllldddds: Kore.Rules2014!
  - That topology would take 92 compute-years to exhaust. Or, 1% every 338 days.
  - For 100,000 users with 9 history records, even if only 1% use this pattern, **you will average cracking one password every 81 hours**.

KoreLogic SECURITY

Agenda

My Background

Classic Password Cracking

Classic Defenses

Recent Trends

PathWell

Examples

Next Steps

Defenses Need to Evolve

- We need to add a new dimension to password strength enforcement.
- Rules like minimum length, minimum character sets required, no dictionary words, etc are still needed.
- But we also need a way to prevent users from gravitating towards the same password patterns (topologies) and overusing them.

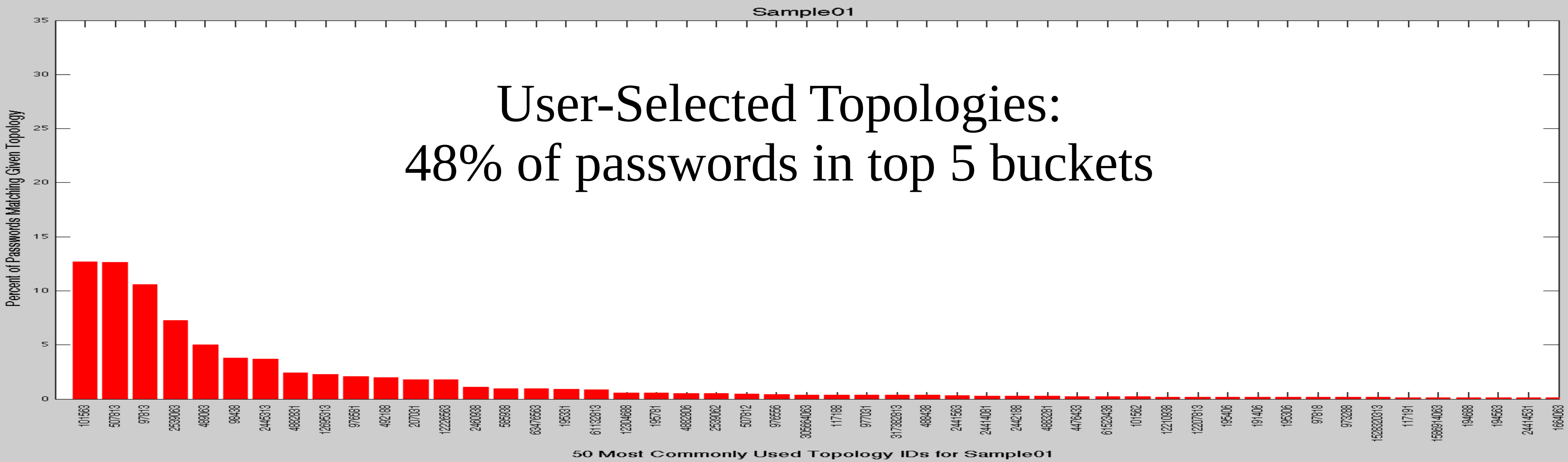
Topology-Related Defenses

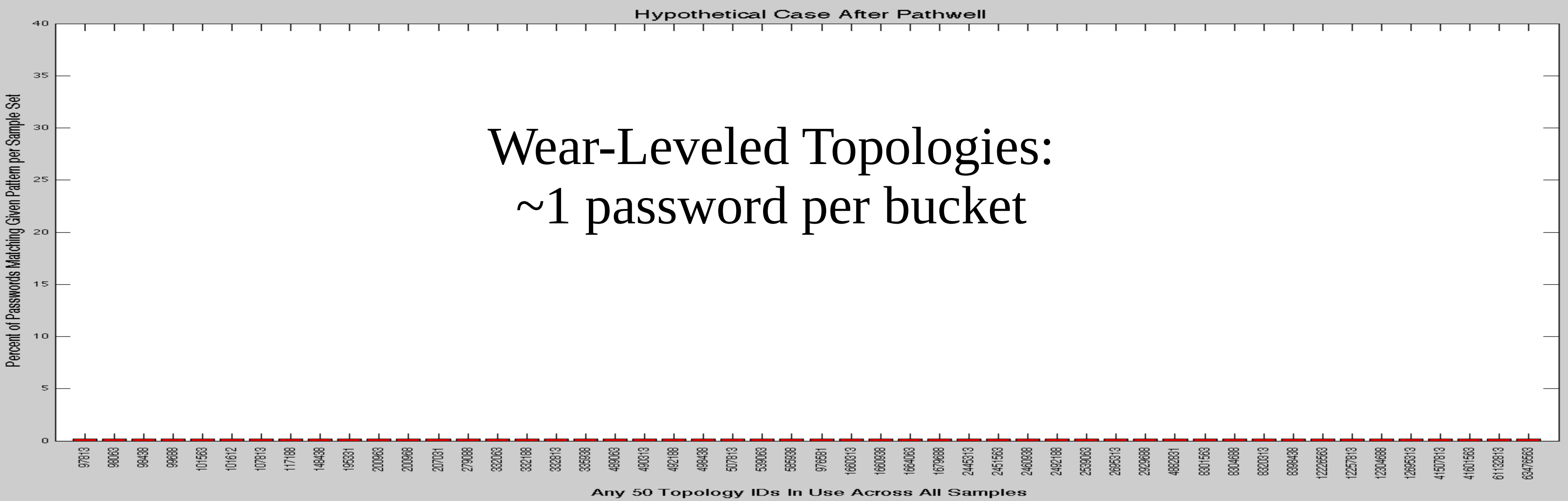
What are some ways we could use this knowledge to level the playing field?

- Blacklist the most common, predictable topologies.
- Don't allow multiple users to stack up on the same topology – force them to spread out. "Wear-Level" them across the possible topology space.
- Require a minimum topology change between old and new passwords.

The primary cost of these is keyspace reduction.

PathWell: Password Topology Histogram Wear-Leveling





Topology Wear-Leveling Effectiveness

How much does topology wear-leveling increase the attacker's work-factor?

- Attacker's work-factor thought of as “**work needed to get the same percentage of cracks**” or “**cracks for the same work.**”
- **Best-case** (fully random topologies): **6 orders of magnitude more work** (one million times as long to get the same number of cracks, or one millionth as many cracks in the same time spent).
- **Worst-case** (attacker knows and goes after only those topologies in use): **2-3 orders of magnitude more work**.
- **Realistic case** (topologies not fully random, attacker makes educated guesses): **4-5 orders of magnitude more work**.

Minimum Topology Change

- Without wear-leveling, a user with password 'Kw#46_Ya' is most likely to set their next password to (say) 'Kw#47_Ya'
- Likewise, with wear-leveling, that user would likely chose 'Kw#46_YA' – the smallest allowable topology change.
- So: the attacker who knows what a user's password topology was in the past, should search the topologies that are “nearest” to it.
- The KoreLogicRulesReplaceNumbers ruleset published back in 2010 can easily crack these variations.

Measuring Topology Change: Levenshtein Distance

- "...[T]he Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertion, deletion, substitution) required to change one word into the other."
  - http://en.wikipedia.org/wiki/Levenshtein_distance
    - Michael Scott
- Sometimes also referred to as "edit distance."
- kitten → **m**itten = 1
- abounds → abound**ed** = 2
- des**s**ert → desert = 1

Measuring Topology Change: Levenshtein Distance

For our examples earlier:

- Kw#46_Ya → Kw#4**7**_Ya
  ulsddsul → ulsddsul = Lev distance 0
- Kw#46_Ya → Kw#46_Y**A**
  ulsddsul → ulsddsu**u** = 1
- P4ssword13! → P4ssword**s**13!
  udllllllldds → udlllllll**l**dds = 1
- P4ssword13! → P**@**ssword1**23**
  udllllllldds → u**s**lllllldd**d** = 2

KoreLogic SECURITY

Cost of Topology-Related Defense: Keyspace Reduction

Don't blacklisting and topology wear-leveling reduce the keyspace that an attacker would have to test for valid passwords?

How much does this keyspace reduction help the attacker?

Cost of Topology-Related Defense: Keyspace Reduction

- **Blacklisting**: For 8-character, 4-charset passwords, there are 4^8, or 65,536 topologies. 100 of them is less than 0.2% of the keyspace. That is a trivial cost and we should gladly pay it. (The cost drops for longer passwords, too.)
- **Forcing unique topology use:** has the downside that the odds that any one randomly selected topology will contain a password go *up*.
  - This effect is worse for larger user populations.
  - However, this is vanishingly small compared to the cost of, say, 5-10% of all users using a single topology that the attacker can easily guess.

PathWell

Developed a PAM module that implements (all optional, administrator-controlled):

- Auditing
- Blacklisting
- Maximum use-count
- Minimum Levenshtein distance

Developed and tested on multiple Linux distributions; not yet tested on any other OS's with PAM support.

PathWell Audit Mode

Audit mode:

- Each time a password is changed, increment a counter for that password's topology.
- Usage counters are not decremented when a password is changed (history lasts forever).
- Useful “standalone” (without enforcement) in order to quantify the problem in a given enterprise.
- Historical data is used by use-count enforcement.
- This DB is sensitive! An attacker who captures it gets some nice hints.
- Current implementation can track topologies up to 29 characters long.

PathWell Enforcement Mode

Enforcement mode option: blacklisting

- Do not allow any user to set a password that uses a known-overused topology.
- We compiled a list of the topologies that we see recur between different enterprise networks.
- Administrators can replace or augment our default list with their own (enabling audit mode can help build up a local, organization-specific list).
- Can also be used to enforce minimum-complexity requirements (blacklist all topologies that do not use 4 of 4 character classes, etc).

PathWell Enforcement Mode

- Note, blacklisting is not enough!
  - If users are just denied their top 100 overused choices, they will probably make similar choices about what to switch to instead.
  - We call that herding, and it is bad... in the long run, attackers just need to learn and adapt to the next-top-100 topologies and start over.
  - Instead, we want mechanisms to not herd users in a group, but rather, shoo them and disperse them more widely across the possible topology space.

PathWell Enforcement Mode

- Note, blacklisting is not enough!
  - If users are just denied their top 100 overused choices, they will probably make similar choices about what to switch to instead.
  - We call that herding, and it is bad... in the long run, attackers just need to learn and adapt to the next-top-100 topologies and start over.
  - Instead, we want mechanisms to not herd users in a group, but rather, shoo them and disperse them more widely across the possible topology space.
- ...But it is better than nothing. You don't have to run faster than the bear...

KoreLogic SECURITY

PathWell Enforcement Mode

Enforcement mode option: maxuse

- Requires that Audit Mode is enabled.
- Set the maximum number of passwords that can use any given topology.
- Typically set to 1 (each password must use a unique topology... until exhaustion/rollover and admins increment it to 2, etc).
- If maxuse=1, then an attacker who bruteforces a topology will score at most one plaintext.

KoreLogic SECURITY

PathWell Enforcement Mode

Enforcement mode option: minlev

- PathWell's minlev enforcement compares a user's old password's topology to the requested new one.
- minlev=1: new password's topology must not be the same as the old. For a 10-char password, there are 30 topologies of the same length of Lev distance 1 for the attacker to target.
- minlev=2: new topology must be at least two changes away from the old. For a 10-char password, there are 405 possible 10-char topologies that are 2 Lev distance away (and more if the length is changed).
- This does not need audit mode to be enabled.

KoreLogic SECURITY

User Reception

Will users revolt?

- Any new control that adds work for them will be resisted.
- Could be mitigated by user-hinting and training (which have their own costs).
- We need some test beds to figure out things like:
  - How many tries does it take the average user to succeed in creating a new password?
  - Which combination of options (blacklist, minlev, etc) provides the most security for the least user burden?
- Any volunteers?

CMIYC Experiment

Pretty math is one thing; how about a real test?

- Crack Me If You Can 2013 included some PathWell-related experiments.
  - “Company1” - “Company5”: non-PathWell-related collections of password hashes (~85% of the total contest hashes).
  - “Company6”: ~10k hashes following the merged distribution from our 8 enterprise samples (a baseline control).
  - “Company7”: Wear-Leveled but no blacklisting. ~10k unique topologies used once each-starting with the most popular and radiating outward.
  - “Company8”: Wear-Leveled using randomly distributed topologies.

KoreLogic SECURITY

CMIYC Experiment

These were then used for different hash types:

Length	Hash Type	% of Length-N
8	UNIX DES crypt	50
8	Salted Sha1 ({SSHA})	25
8	FreeBSD MD5 (1)	25
9+	NTLM (NT MD4)	75
9+	Unsalted Sha1 ({SHA})	25

Note: We know that 9 char is still fatally short for NTLMs and unsalted SHA1 – we used them to keep the contest engaging.

CMIYC Experiment Results

"What the hell was Company8 doing? We can't crack any of them!"

CMIYC Experiment Results

Pro class teams' merged unique cracks

Hash Type	Company6 (control)	Company7 (unique, predictable)	Company8 (unique, random)
NTLM	1368	101	7
NSLDAP (SHA1)	181	8	0
UNIX DES	30	1	0
Salted SHA1	9	1	0
FreeBSD MD5	2	0	0

Street class teams' merged unique cracks

Hash Type	Company6	Company7	Company8
NTLM	648	53	0
NSLDAP (SHA1)	209	0	0
UNIX DES	9	1	0
Salted SHA1	2	0	0
FreeBSD MD5	0	0	0

Agenda

- **My Background**
- **Classic Password Cracking**
- **Classic Defenses**
- **Recent Trends**
- **PathWell**
- **Examples**
- **Next Steps**

Example Configurations

Example /etc/pam.d settings:

- Default:

```
password   required   pam_cracklib.so difok=2 minlen=8 dcredit=2 ocredit=2 retry=3
password   required   pam_unix.so try_first_pass use_authtok nullok sha512 shadow
password   optional   pam_permit.so
```

- Audit mode:

```
password   required   pam_cracklib.so difok=2 minlen=8 dcredit=2 ocredit=2 retry=3
password   required   pam_unix.so try_first_pass use_authtok nullok sha512 shadow
password   optional   pam_pathwell.so mode=monitor use_authtok
password   optional   pam_permit.so
```

- Blacklist mode:

```
password   required   pam_cracklib.so difok=2 minlen=8 dcredit=2 ocredit=2 retry=3
password   required   pam_pathwell.so mode=enforce use_authtok blacklist
password   required   pam_unix.so try_first_pass use_authtok nullok sha512 shadow
password   optional   pam_permit.so
```

- Maxuse mode:

```
password   required   pam_cracklib.so difok=2 minlen=8 dcredit=2 ocredit=2 retry=3
password   required   pam_pathwell.so mode=enforce use_authtok maxuse=1
password   required   pam_unix.so try_first_pass use_authtok nullok sha512 shadow
password   optional   pam_pathwell.so mode=monitor use_authtok
password   optional   pam_permit.so
```

- Minlev mode:

```
password   required   pam_cracklib.so difok=2 minlen=8 dcredit=2 ocredit=2 retry=3
password   required   pam_pathwell.so mode=enforce use_authtok minlev=2
password   required   pam_unix.so try_first_pass use_authtok nullok sha512 shadow
password   optional   pam_permit.so
```

Example Error Messages

- Some example errors from various enforcement modes (syslogged, not visible to the user):

  ```
  Nov 13 22:36:50 foo passwd[12416]: pam_pathwell(passwd:chauthtok): Release='0.6.0'; Library='1:0:0'; Module='0:1:0'; PamFlags='0x00002000'; Mode='enforce'; User='testuser'; Error='The topology associated with the chosen password has been blacklisted.';
  ```

  ```
  Nov 13 22:37:06 foo passwd[12418]: pam_pathwell(passwd:chauthtok): Release='0.6.0'; Library='1:0:0'; Module='0:1:0'; PamFlags='0x00002000'; Mode='enforce'; User='testuser'; Error='The topology associated with the chosen password would exceed the maximum allowed use count.';
  ```

- ```
  Nov 13 22:37:45 foo passwd[12420]: pam_pathwell(passwd:chauthtok): Release='0.6.0'; Library='1:0:0'; Module='0:1:0'; PamFlags='0x00002000'; Mode='enforce'; User='testuser'; Error='The topology associated with the chosen password does not meet the minimum required Lev distance.';
  ```
  ```

Agenda

- My Background
- Classic Password Cracking
- Classic Defenses
- Recent Trends
- PathWell
- Examples
- **Next Steps**

PathWell: Next Steps for the Project

- Need to test / gather data with real users
  - Audit mode: does our current list of the worst topologies hold up for other user populations?
  - When the different enforcement modes are enabled, how many tries does it take the user to successfully set a new password?
  - Study user hinting options.
  - Doing a usability study this summer.
  - Again, any volunteers to do a pilot deployment?

PathWell: Next Steps for the Code

- Open source the current PAM implementation – later this summer (granting a license for our pending patent).
- Support for enterprise environments
  - Windows Active Directory!
  - Enterprise LDAP platforms
  - Other UNIX (Non-Linux) PAM systems
  - Large web applications / websites?
  - NIS
- Non-password applications? PINs?

PathWell: Next Steps for Attackers?

How will attackers – cracking tools, techniques – adjust and adapt to PathWell?

That's all folks

Questions?

Hank Leininger <hlein@korelogic.com>
D24D 2C2A F3AC B9AE CD03 B506 2D57 32E1 686B 6DB3

PathWell Project <pathwell-project@korelogic.com>
42E7 8319 21F3 01C8 2D72 A591 35EA 3CC7 502D 942F

Thanks to:

Klayton Monroe	Sean Segreti
Shawn Wilson	Mick Wollman
BITSys	DARPA!
CMIYC Teams	Hashcat / JTR

Other Reading

- https://blog.korelogic.com/ ← will post links to this talk soon
- @CrackMeIfYouCan on Twitter
- CMIYC contest sites; past years have teams' writeups:
  - http://contest-2014.korelogic.com/
  - http://contest-2013.korelogic.com/
  - http://contest-2012.korelogic.com/
  - http://contest-2011.korelogic.com/
  - http://contest-2010.korelogic.com/
- My coworker Rick Redman has given a number of talks about advanced password cracking techniques:
  - Passwords13: http://www.youtube.com/watch?v=5i_Im6JntPQ
  - ISSA: http://infosec-summit.issa-balt.org/html/2010_agenda.html

  Rick goes into detail about advanced cracking techniques, various rules we've written for different tools & how to write your own.
- An interesting study about studying password selection: "On The Ecological Validity of a Password Study": http://cups.cs.cmu.edu/soups/2013/proceedings/a13_Fahl.pdf