Supercharged John the Ripper Techniques

Austin OWASP

Spring, 2011

Rick Redman - KoreLogic
Who am I:
Rick Redman – Senior Security Consultant – Penetration Tester

Bio: During my 11 years as a security practitioner, I have delivered numerous application and network penetration tests for a wide range of Fortune 500 and government clients. I serve as KoreLogic's subject matter expert in advanced password cracking systems. I present at a variety of security forums such as ISSA Chapters and AHA (Austin Hackers Anonymous) and provides technical security training on topics such as web application security. I has served as a member of a penetration testing tiger team supporting Sandia National Laboratories. I am a graduate of Purdue University with a degree in Computer Science in the CERIAS/COAST program taught by Gene Spafford.
Who is KoreLogic:

- An expert-based information security and IT risk management firm.
- Serve Fortune 500 and Government clients.
- 500+ security engagements delivered.
- Invited speakers: OWASP, Shmoocon, CEIC, SIM, ISSA, DoD, Universities
- Creator: “Crack Me If You Can” password cracking contest at DEFCON
- Privately held and founder-operated allow us to practice a quality- and client-first approach.
Today’s goal:

Show default password cracking methods/tools. Pros/Cons of each.

Show patterns used by users in environments that enforce password complexity (patterns observed from over 3.1 million hashes cracked by KoreLogic).

Improve the methods/rules used to crack passwords, in order to crack large amounts of complex passwords.

Give advice on how to create complex passwords that aren’t based on known patterns.

  – **OWASP**: Show previous errors done by web application developers that exposed passwords.
10 Users choose bad passwords
20 SysAdmins put in place password complexity rules
30 GOTO 10

Corporate users are becoming more and more aware of the importance of stronger passwords. They are **forced** to follow password requirements.

- Tools used to crack passwords, need to be made more aware of the patterns used by users who are forced into meeting password complexity rules. (Currently, most tools do not do this.)

- Development of current password cracking tools does not revolve around patterns and wordlists.

Instead seems to be concentrated on more formats, brute forcing, using the “cloud”, distributing work, GPU cards, etc.
10 Users choose bad passwords
20 Web sites on the Internet allow this to occur
30 Web sites get hacked – Users' passwords are cracked
40 Nothing changes – GOTO 10

- Developers are not aware of proper methods of storing password hashes
- The Internet stays “unsafe” and no progress is made in the web application world to educate or **protect** our users.
- Applications continue to allow users to choose crappy passwords.
- Applications continue to not allow security conscious users to protect themselves (See: List of websites that don't allow special characters in their passwords, or limit the length allowed)
<table>
<thead>
<tr>
<th>Type</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaintext</td>
<td>badpass</td>
</tr>
<tr>
<td>MD2</td>
<td>9C5B091C305744F046E551DB45E7C036</td>
</tr>
<tr>
<td>MD4</td>
<td>640061BD33AA12D92FC40EA87EA408DE</td>
</tr>
<tr>
<td>MD5</td>
<td>F1BFC72887902986B95F3DFDF1B81A5B</td>
</tr>
<tr>
<td>SHA-1</td>
<td>AF73C586F66FDC99ABF1EADB2B71C5E46C80C24A</td>
</tr>
<tr>
<td>LM</td>
<td>4CF3B1913C3FF376</td>
</tr>
<tr>
<td>NT</td>
<td>986CA892BEAB33D1FC2E60C22EC133B7</td>
</tr>
<tr>
<td>MySQL323</td>
<td>0AFDA7C85EE805C2</td>
</tr>
<tr>
<td>MySQLSHA1</td>
<td>229749C080B28D3AEFAB78279C4668E6E12F20FA</td>
</tr>
<tr>
<td>Cisco PIX</td>
<td>RtJk8qcK DPR.2D/E</td>
</tr>
<tr>
<td>VNC Hash</td>
<td>DAD3B1EB680AD902</td>
</tr>
</tbody>
</table>
“RockYou” - passes stored in plain-text in database. SQL Injection. Millions of plain-text Facebook/MySpace passwords obtained.

“Gawker” - Database obtained – ~750,000 User password hashes obtained (Salted DES)

Rootkit.com – Database Backup Obtained - ~58000 User Password Hashes (Raw MD5)

“Xiaozhi” - Database Attacked – 1.06 Million Password Hashes (SHA Variant)

Unnamed Financial Site – ~5 Million User Password Hashes obtained (raw MD5)
Hash EXchange!

About
Here you can browse cracked hashes, download left hashes and upload found hashes.

Lists

<table>
<thead>
<tr>
<th>list</th>
<th>download</th>
<th>type</th>
<th>found</th>
<th>left</th>
<th>total</th>
<th>progress</th>
<th>dictionary</th>
<th>comments</th>
<th>upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>airsoft</td>
<td></td>
<td>md5</td>
<td>7,006</td>
<td>141</td>
<td>7,147</td>
<td>98.03%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>carders</td>
<td></td>
<td>md5(md5(pass).salt)</td>
<td>2,214</td>
<td>6,211</td>
<td>8,425</td>
<td>26.28%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fcolmpia</td>
<td></td>
<td>mysql64</td>
<td>3,726</td>
<td>639</td>
<td>4,356</td>
<td>85.54%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>freehackforumall</td>
<td></td>
<td>md5(md5(pass).salt)</td>
<td>25,335</td>
<td>16,390</td>
<td>41,725</td>
<td>60.72%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gamepasson</td>
<td></td>
<td>md5</td>
<td>2,944</td>
<td>38</td>
<td>2,982</td>
<td>98.73%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gawkers</td>
<td></td>
<td>des</td>
<td>674,690</td>
<td>69,174</td>
<td>743,864</td>
<td>90.7%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hellbound</td>
<td></td>
<td>md5</td>
<td>5,949</td>
<td>277</td>
<td>6,226</td>
<td>95.55%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inbookmark</td>
<td></td>
<td>md5</td>
<td>2,101</td>
<td>229</td>
<td>2,330</td>
<td>90.17%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jaillords</td>
<td></td>
<td>md5</td>
<td>2,348</td>
<td>94</td>
<td>2,442</td>
<td>96.15%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>challengers</td>
<td></td>
<td>md5</td>
<td>3,884,294</td>
<td>1,524,125</td>
<td>5,408,419</td>
<td>71.82%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>phpbb</td>
<td></td>
<td>md5</td>
<td>186,127</td>
<td>3,540</td>
<td>189,667</td>
<td>98.13%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rootkit</td>
<td></td>
<td>md5</td>
<td>54,372</td>
<td>4,303</td>
<td>58,675</td>
<td>92.67%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scrollwars</td>
<td></td>
<td>md5</td>
<td>11,570</td>
<td>369</td>
<td>11,939</td>
<td>96.91%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tntvillage</td>
<td></td>
<td>md5</td>
<td>36,154</td>
<td>1,681</td>
<td>37,835</td>
<td>95.56%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>twilight-disorder</td>
<td></td>
<td>md5</td>
<td>5,079</td>
<td>98</td>
<td>5,177</td>
<td>98.11%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wsgifts</td>
<td></td>
<td>md5</td>
<td>2,798</td>
<td>70</td>
<td>2,868</td>
<td>97.56%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xiaozhi</td>
<td></td>
<td>sha256</td>
<td>1,068,680</td>
<td>729,782</td>
<td>1,798,462</td>
<td>59.42%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Theory (OWASP)

Internet-based web-application passwords always suck. OWASP should fix this.

Passwords on the web are worse because sites on the Internet do not:
   1) Force password complexity
   2) Require/Force passwords changes/rotation
   3) Educate their users

Password cracking tools are mostly tested to work against these subpar passwords – and not against passwords chosen against password complexity (and rotation) policies

Password cracking tools/techniques must be improved in order to crack the “stronger” passwords that are created against complexity rules.

Once web applications on the Internet start encrypting their hashes with stronger formats, and start educating/forcing their users to choose better passwords – the default tools will not work as well – and their users will be “safer”..... for now.
Password Cracking Tools:

1) John the Ripper (Our preference - Today's Topic)
2) HashCat / OCLHashCat / OCLHashCat+ (Recommended Tools)

3) SAMInside - Dictionary section has extremely basic rules (Approx 10) Prepend 1-2 characters - Append 1-2 characters.
4) L0phtCrack 6 - "Strong Password Audit - "common modifications" consists of Prepending and/or Appending 2 characters.
5) ophcrack - Rainbow Tables Based - Brute Force
6) PasswordsPro - Supports the MOST formats of all tools Very slow to load input files with multiple passwords Actually has a "Rules.txt" file very simular to John the Ripper - these rules are also almost as good as John's default ruleset. Costs Money. (Approx $54 USD).
7) Cain/Abel - Free - Has really basic rules (reverse, Double, Case Subs, 2 numbers append, l33t rules)
HashCat / OCLHashCat /OCLHashCat+ (Recommended Tools)

- Hashcat is “closed source” (but free)
- Automatically takes advantage of all cores/cpus (great for multi-core systems)
- Has “rules” that are semi-compatible with John the Ripper
- Under constant development
- Very active IRC/Message-board based user-base
- Supports large amounts of formats (NTLM, SHA, MD5)
- Not as “user friendly” as other tools – expect a learning curve

- OCLHashCat uses GPU cards (ATI / NVidia) and can be used to build incredibly powerful systems for very little $$$

- The HashCat team won KoreLogic’s 2010 “Crack Me If You Can” password cracking contest at DEFCON
John the Ripper (JtR):

http://www.openwall.com/john/
http://www.openwall.com/john/doc/

Mailing List:
http://marc.info/?l=john-users&r=1&w=2

"John the Ripper is a fast password cracker, currently available for many flavors of Unix (11 are officially supported, not counting different architectures), Its primary purpose is to detect weak Unix passwords."

It is free, it is open source, it is constantly under development.

A team based around JtR came in 3rd and 4th place in “Crack Me If You Can” password cracking contest at 2010 DEFCON.
Examples of Usage:

```bash
# john /etc/shadow
# john --wordlist=password.lst --rules passwd
# john --show passwd

Loaded 17461 password hashes with no different salts (NT)
test (username1)
password (username2)
password1 (username3)
123456 (username4)
qwerty (username5)
baseball (username6)
```
Yes it cracks passwords, but how?

1) Uses a wordlist (supplied with the tool).

2) Uses a wordlist combined with "rules" that manipulate the wordlist.

3) Brute forces password possibilities based on statistics generated by the primary developer (and older tools).

These are roughly the same ideas that all password cracking software packages use.
Problems with these methods:

1) Default wordlist is small/outdated/mostly based on statistics of extremely weak passwords.

Wordlists for all tools are not based on recent password statistics. Also, not based on passwords used in “Corporate” environments.

**OWASP:** The default wordlists *do* work pretty good on web users

Publicly available wordlists are also not based on recent password statistics.

- Klingon? Swahili? Esperanto? No one uses these.

Even if you use real password statistics, from where? Do people on Facebook choose better or worse passwords than internal corporate networks? (Example: RockYou).

- **OWASP:** Find the 'RockYou' list – it will crack TONS of Internet based hashes
Problems with these methods (cont):

2) Rules are based upon statistics gained from a limited data pool. This data is old/outdated. Appear to be based on techniques used by users in the 1990s. Users today are choosing much more complex patterns. Users are forced to choose more complicated passwords because of password policies.

3) The Brute Force file (all.chr) is based on outdated passwords lists. All.chr contains statistics about letters/letter combinations used in previous passwords. These statistics will not quickly match your password statistics for your users in the 2010s.

So, let's crack some passwords…
About today's examples:

Password file used: pwdump.txt (Format NTLM – From an Windows Active Directory)

This is a file containing 32883 valid hashes from a mythical single company.

This company uses strict password policies that enforce complexity rules.
   Upper Case - Lower Case - and Numbers are required.

   Special Characters are highly encouraged by security staff (but not enforced).

**OWASP:** Does your web-site do this? Why not...?
What can 'john' do by default:

```bash
# john --format:nt -w:password.lst pwdump.txt
Loaded 32883 password hashes with no different salts (NT)
```

Example of cracked passwords:

- august
- backup
- baseball
- blowfish
- bluesky
- austin
- bridge
- change
- enterprise
- football
- front242
- goldfish
- health1
- holiday
- london
- looney
- password
- patriots
- research
- security
- services
- station
- stupid
- sunshine
- watson
- winter
- yellow
- welcome

Guesses: 29 unique passwords found (101 actual accounts obtained, multiple accounts shared these passwords)
Now with John's rules (notice the --rules):

```
# john --rules --format:nt -w:password.lst pwdump.txt
```

<table>
<thead>
<tr>
<th>Abcd1234</th>
<th>Abigail7</th>
<th>Alexander5</th>
<th>Allison9</th>
<th>Anthony9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aragorn3</td>
<td>Arsenal1</td>
<td>Arsenal4</td>
<td>Asdf1234</td>
<td>Asterix9</td>
</tr>
<tr>
<td>Autumn1</td>
<td>Baseball3</td>
<td>Baseball6</td>
<td>Beaches1</td>
<td>Beautiful2</td>
</tr>
<tr>
<td>Belgium2</td>
<td>Belmont7</td>
<td>Benjamin3</td>
<td>Birthday6</td>
<td>Blessed1</td>
</tr>
<tr>
<td>Bonjour1</td>
<td>Bonjour2</td>
<td>Bonjour3</td>
<td>Dallas1</td>
<td>Dallas2</td>
</tr>
<tr>
<td>Dallas6</td>
<td>Passw0rd</td>
<td>Password1</td>
<td>Password2</td>
<td>Password3</td>
</tr>
<tr>
<td>Stingray2</td>
<td>Stingray?</td>
<td>Zachary2</td>
<td>april9</td>
<td>austin1</td>
</tr>
</tbody>
</table>

guesses: 272 unique **new** passwords found (846 actual total accounts obtained)
Now with --single (a more advanced set of rules based around the username).

```
# john --single --format:nt pwdump.txt
```

(Notice no wordlist/dictionary)

guesses: 1361 unique new passwords found (1462 actual total accounts obtained)

Munich09 Exchangeftp Computer55 Summer55
London999 Orange123 mcafee Project1
Dublin89 citrix2000 aug1999 Corporate1!!
# john --rules:single --format:nt -w:password.lst pwdump.txt
(This attempts passwords with rules based off of the wordlist 'password.lst' – This is a new “trick” not documented anywhere)

password  Yankees9  Sydney12  London33  baseball  Clippers9
London12  Michael22  football  Asterix9  Syntel12  Report22
sunshine  History9  Mercer12  Australia22august  Holiday9
Cayman12  Test0123  welcome  Patriots9  Scudder23Account09
backup  Redfish9  Munich23  security  Surfing9  Munich13
winter  Dolphins9  Market13  Dallas23  Tiffany9  Dallas13
London23  london  Trinity9  London13  Paris123  ireland123
Ireland09  London444  France888  newyork1!  hello123#
qwerty1!  Munich23  abcd1234%

guesses: 1283 unique new passwords found (3645 actual new accounts obtained. Multiple shared passwords)
After all 4 steps are run, in this example we have:

3645 actual accounts with logins and passwords obtained. 'john' reports: 9980 password hashes cracked, 38921 left

This is because 6337 accounts had blank passwords (they were disabled accounts).

This is 20% of the entire password file cracked in a few short amount of time.

(approx: 120 seconds)
So, what’s the problem? You cracked 20% of the passwords and you are brute forcing the rest of the range. Wont you eventually get 99% done?   \textbf{Answer: Not even close!}

- You have barely scratched the surface of what users are \textit{really} doing to generate passwords.
- You are also wasting CPU cycles by generating passwords that don't meet the known patterns that your users are using.
- You are also trusting the 'rules' inside of John the Ripper in order to discover patterns chosen by users.

What patterns we found using default settings:
1) Adding numbers on the end of passwords.
2) Capitalizing the first letter of each word
3) Adding a ! to the end of a Capitalized word
4) Adding 123
5) Variations on the Usernames (adding specials/numbers/etc).
If you look at the output of:

```
# john  -i:all -stdout | head -n 1000000 | egrep [A-Z] | wc
```

This returns: 21655 passwords with a capital letter.

You will see that of the first 1,000,000 passwords attempted via brute-force, only 21,655 contain capital letters (2%) and 162 of them _start_ with a capital letter. (.01%)

But, of the 2235 unique passwords cracked so far, 1976 contain a capital letter (and 1975 of those 1976, _start_ with that capital letter). This represents 88% of the passwords cracked.

Why not take advantage of this statistic, and dig deeper into wordlists, and rules to crack more passwords?

**OWASP:** If your web site does not require a capital letter, even the most beginner password cracker will destroy your hashes!
Up to this point:  
9980 password hashes cracked, 38921 left

We can assume we will crack more and more passwords using "all.chr":

But we do not know how fast they will crack? So, why risk it?

Instead, using our rules/tricks/tips, KoreLogic was able to obtain the following stats:

43131 password hashes cracked, 5770 left

How did we do it?
By looking for patterns! Examples:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ABla1109</td>
<td>Domi1236</td>
<td>July2006</td>
<td>Novb2009</td>
<td>Sept2010</td>
<td></td>
</tr>
<tr>
<td>AChw0708</td>
<td>Dons0117</td>
<td>July2007</td>
<td>Nove1234</td>
<td>Sept2012</td>
<td></td>
</tr>
<tr>
<td>AaKw2013</td>
<td>Doom2009</td>
<td>July2009</td>
<td>Nove2007</td>
<td>Sept7860</td>
<td></td>
</tr>
<tr>
<td>Abcd1234</td>
<td>Dune2001</td>
<td>July2498</td>
<td>Nove2009</td>
<td>Sepx2009</td>
<td></td>
</tr>
<tr>
<td>Abcz2009</td>
<td>Dyln0202</td>
<td>July6060</td>
<td>Novo2009</td>
<td>Sesp2010</td>
<td></td>
</tr>
<tr>
<td>Abhi1009</td>
<td>EHeh8888</td>
<td>June1984</td>
<td>Novs2009</td>
<td>Sfax2014</td>
<td></td>
</tr>
<tr>
<td>Acac3434</td>
<td>Edin1485</td>
<td>June2007</td>
<td>Novu2009</td>
<td>Shal1234</td>
<td></td>
</tr>
<tr>
<td>Adam1109</td>
<td>Edin2006</td>
<td>June2009</td>
<td>Novv2009</td>
<td>Shaw0709</td>
<td></td>
</tr>
<tr>
<td>Addi0204</td>
<td>Ekim2005</td>
<td>June2012</td>
<td>OOcto2009</td>
<td>Shiu0209</td>
<td></td>
</tr>
<tr>
<td>Addi5678</td>
<td>Ekim2009</td>
<td>June2020</td>
<td>OOdd2233</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What patterns do you see?
Answer: [A-Za-z]{4}[0-9]{4} (Letters 4 times - then 4 Numbers)

The pwdump.txt example had 1575 accounts with a password that met this pattern (647 unique examples)

The most common passwords of which were:
144 Fall2010
139 Sept2010
  80 Octo2010
  49 Nove2010
  16 Augu2010
  12 July2010
  12 Dece2010

See another pattern? (Look at the letters used).
So in the previous example, if we had a rule that appended 4 numbers to the end of a password - We could crack more of these.

The original john.conf does some of these:

- [:c] (?a \p1[lc] Az"[1-9]\0\0" <+ \\
  l Az"19[7-96-0]" <+ >-
  l Az"20[01]" <+ >-
  l Az"19[7-9][0-9]" <+ \\

But, who can read these rules?

So, we have identified a password pattern, and we want to make a rule that will search for all possible combination that fit that pattern.

How do we do it?
We write our own rules in john.conf!

[List.Rules:KoreLogicRulesAppend4Num]
c$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]

The 'c' means - begin each try with a capital letter (remember our stats, 88% of the passwords started with a capital letter).

$[0123456789]$ Means add a 0 or 1 or 2 ... to the END of the string. ($ = END)
Example Output

# more foo.dic
test

# john -w:foo.dic --rules:KoreLogicRulesAppend4Num -stdout
Test0000 Test0001 Test0002 … (etc etc etc)
Test9998 Test9999 test0000
test0001 test0002 test 0003 … (etc etc etc)
test9997 test9998 test9999

Notice: capital 'T' in the first 10000 tries
Notice: lower case 't' in the last 10000 tries
How else can I improve this? Use better wordlists!

With KoreLogicRulesAppend4Num use:

1) All 4 letter words (this is fast)
http://www.justpain.com/ut_maps/wordlists/length04.txt

# john -w:length04.txt --rules:KoreLogicRulesAppend4Num -format:nt pwdump.txt

2) Create you own list of 4 letter words based upon words you've already seen (Remember the months we saw previously? Create a list of those)

3) All 4 letter combinations (aaaa aaab aaac .... zzzy zzzz) (slower)
   9,139,520,000 possible combinations

4) All 4 character combinations (0000 000a 000b ... !!!a !!!b .. @%!% ...)
   (This is really slow - but can be productive)
Types of Wordlists

Remember:

There are lots of other wordlists you can try with this one rule:
1) default wordlist  
2) 3 letter words/combinations  
3) 5 letter words  
4) wordlists from 3rd parties  

With our example pwdump.txt the following command line:
# john -w:4letters.dic --rules:KoreLogicRulesAppend4Num --format:nt pwdump.txt

We cracked an additional 597 unique passwords. (Totaling 1572 new accounts) This took roughly 27 minutes. Would be _much_ faster with the tool ‘hashcat’ or ‘oclhashcat’

Now up to:  11552 password hashes cracked, 37349 left
Additional patterns used by users (and the rules to crack them):

New Pattern: Use of '1234'
The default john.conf adds '123' to the end of every word, but not 1234. And not at all positions.

1234!!@@1234 1234pass !QAZ1234 Pass1234
1234!@# 1234Hlya !Qwe1234 1234!@#$ 1234Hacc
1234paul ##1234## 1234!PA 1234SU$ car1234
Pqrs1234$1234Help. $oct1234 1234*555 1234Qwer !
1234mrs $Work1234 1234qwe 1234Harley !1234Sunny
Smab1234%% 1234+++ 1234Qwerty 1234password

Also, KoreLogic released rules that will also append it at the beginning, middle, end, etc of each word in our wordlist.
Users love using '123' as their numbers:

!Austin123  (summer123  123bali  1234Wipro@
123Austin$  123!Saints  123-Saints  123dani
123August$  !Elaine123  1!Brawn123  123dini
123August/  123Clipper!  123$$$  Photon123!!
Ferrari123  123roka123  August@123 Clippers#123
Jordan123  1samuel$123  August_123  $Dipesh123
Austin#123  whatuwant@123  whatwewant@123

In [List.Rules:Wordlist] - just add a line that says: $1$2$3
Current Year: Lots of users will use the current year as their number:

!Jan2010  2010!!  Work2010aha  May2010mjk  Sep2010mjk
020102mc  Alps2010!  ck2010ck  cl2010qt  Sept2010x
020104jo  Augu2010$  slm2010md  co2010je
2010!  Winter2010a  Jam2010sic  Pal2010mine

Same idea as previous rule, place 2010 at the beginning, middle, end.


i0[2]i1[0]i2[1]i3[0]
i1[2]i2[0]i3[1]i4[0]
i2[2]i3[0]i4[1]i5[0]
i3[2]i4[0]i5[1]i6[0]
i4[2]i5[0]i6[1]i7[0]
i5[2]i6[0]i7[1]i8[0]
i6[2]i7[0]i8[1]i9[0]
# more foo.dic
abcd

# john -w:foo.dic -stdout --rules:KoreLogicRulesAdd2010Everywhere
2010abcd
a2010bcd
ab2010cd
abc2010d
abcd2010

Use with wordlists not containing numbers. Just letters and specials.
Lets extrapolate this further, look at these sample passwords:

!Mar2010  $Aug2010  aug2010  1dec2010  @nov2010  $Mar2010
Ljun2010  @Nov2010

These are based on Month/Year combinations.

If it is October of 2010, what do you think most of these passwords are going to be?
What about:
evg2010!  Happy2010@  Nove2010  #password2010@

Hint: Use this with wordlists of letters only. 1-5 letters work best. (at first)
Why? Because we are supplying the numbers and special characters for you.

Prepending Years works as well:

2010Jacob  2010ly!!  2001MARK2010!!  2010#dec  2001Papa
2010tiny  2001JUL  2001andy  2010!Nove  2010+november
2010oct*T  2001JUN  2010September  2010Toyota
What about all these months we keep seeing? They aren't always at the beginning and end. They can be in the middle.

<table>
<thead>
<tr>
<th>Month</th>
<th>Month</th>
<th>Month</th>
<th>Month</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 1008</td>
<td>Oct 05</td>
<td>Jan 1191</td>
<td>May 1969</td>
<td>$\text{Oct06!}$</td>
</tr>
<tr>
<td>Sep 06</td>
<td>Cop Sep 1$</td>
<td>Jan 1985</td>
<td>May 1976</td>
<td>$#\text{Oct999}$ $\text{Sep123}$</td>
</tr>
<tr>
<td>Jan 1993</td>
<td>Oct $\text{Oct}$</td>
<td>Sep 10</td>
<td>Sep 05</td>
<td>Aug Sep 001</td>
</tr>
<tr>
<td>Jan</td>
<td>19835 May</td>
<td>Amy Oct 1</td>
<td>409 Sep 89</td>
<td>01 Jan 184</td>
</tr>
<tr>
<td>Sep 87%</td>
<td>ACSep*123</td>
<td>Zq$\text{Sep9}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What about passwords that end with Month and/or Dates?

<table>
<thead>
<tr>
<th>Month</th>
<th>Month</th>
<th>Month</th>
<th>Month</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2006</td>
<td>2006% May</td>
<td>BN MAY2</td>
<td>MAY77</td>
<td>Simon@May08</td>
</tr>
<tr>
<td>May 2006</td>
<td>Dan &amp; May 1</td>
<td>1971 May 05</td>
<td>Jane May 01</td>
<td>13126 May</td>
</tr>
<tr>
<td>Cap May 7</td>
<td>1 May 06</td>
<td>Jkm May 22</td>
<td>14082 May</td>
<td>1985 May 7</td>
</tr>
<tr>
<td>2$ May 20</td>
<td>vd May 29</td>
<td>1976 May</td>
<td>1989 May 9</td>
<td>*09 May 08</td>
</tr>
<tr>
<td>2006! May</td>
<td>1990 May 2</td>
<td>MAY07</td>
<td>Emily May 07</td>
<td></td>
</tr>
</tbody>
</table>
What about passwords that have the _whole_ month in them:

<table>
<thead>
<tr>
<th>January!12006</th>
<th>February2008=</th>
<th>April$0410</th>
<th>August/2008</th>
<th>October&amp;11</th>
</tr>
</thead>
<tbody>
<tr>
<td>January!2006</td>
<td>March#16</td>
<td>April*26th</td>
<td>August132008!</td>
<td>October/2010$$</td>
</tr>
<tr>
<td>January#2006</td>
<td>March#2008</td>
<td>June%&amp;2011</td>
<td>SeptembeR2010</td>
<td>december97</td>
</tr>
<tr>
<td>January101994</td>
<td>March#3164</td>
<td>June!22004</td>
<td>Septem2010*</td>
<td>december98</td>
</tr>
<tr>
<td>January2010!</td>
<td>March031708</td>
<td>June/2007</td>
<td>September**123</td>
<td>december@01</td>
</tr>
<tr>
<td>February!2007</td>
<td>March142010</td>
<td>August*2010</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Also: 4 letter months

<table>
<thead>
<tr>
<th>OCTO!!2</th>
<th>Octo!200</th>
<th>Octo2010!</th>
<th>Octob#10</th>
<th>Nove2005!</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCTO!01</td>
<td>Octo1957</td>
<td>Octo2010!!</td>
<td>octoocto</td>
<td>Nove2005*</td>
</tr>
<tr>
<td>OCTO!2</td>
<td>Octo1975</td>
<td>Octo2010$</td>
<td>NOVE!20</td>
<td>Nove2005-</td>
</tr>
<tr>
<td>OctO2008$</td>
<td>Octo2**9</td>
<td>Octob!!05</td>
<td>Nove.2008</td>
<td>OctOct09</td>
</tr>
<tr>
<td>Octo200$</td>
<td>Octob!08</td>
<td>Nove.2010</td>
<td>Octo!!2005</td>
<td>Octo200%</td>
</tr>
<tr>
<td>Octob!23</td>
<td>Nove002.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Days of the Week

Also: Days of the week:

MONDAY.  Monday#  Monday0915$  Friday*15  
Thursday2.0  MONDAY/  Monday#01  Tuesday%$888  
Friday.56  Thursday99=  MONDAY0  Monday#123  
friday@2010  Thursday=01  Monday!23  TUESday180105  

[List.Rules:KoreLogicRulesPrependDaysWeek]
i0[mM]i1[oO0]i2[nN]i3[dD]i4[aA4@]i5[yY]
i0[tT]i1[uU]i2[eE3]i3[sS]i4[dD]i5[aA4@]i6[yY]
i0[wW]i1[eE]i2[nN]i3[dD]i4[sS]i5[dD]i6[aA4@]i7[yY]
i0[tT]i1[hH]i2[uU]i3[rR]i4[sS]i5[dD]i6[aA4@]i7[yY]
....
Users love numbers!

By default john.conf has multiple rules that add numbers to wordlists. By default the following passwords can be cracked:

aaron698  windsor003  Aimhigh300  Austin934  Buster172
aaron699  welcome222  Alexis333  Austin958  Caitlin442
ababy420  welcome456  Alliecat789  Austin987  Accounting785

and

ABcd4567   Abby0077  Amanda5878  ABcd5678  Abby0206
Antigua4444  ABcd6789  Abby0217  Antionette0824

By adding 3 or 4 numbers to the end of a wordlist. But john.conf does not do this for all 3 or 4 numbers. Just a subset
So we need rules that do the **complete** list (as seen earlier in presentation):

* [List.Rules:KoreLogicRulesAppend4Num]  
  c$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]

and

* [List.Rules:KoreLogicRulesAppend3Num]  
  c$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]$[0123456789]

Sample output (for Append4Num):

Abcd0000 Abcd0001 Abcd0002 Abcd9998 Abcd9999  
abcd0000 abcd0001 abcd0002 abcd9998 abcd9999
Users Love Numbers

Prepending with 2 Numbers (notice the first capital letter):

00Brandon 01Bahadur 01Jaguars 01Otavalo 99Gretzky 00Presque
01Bigturd 01Jarrett 12Chucker 99Matthew 00Shootme 01Bilusha
01Kokonut 12Cowboys 99Monster 00Tiffany 01Buttons 01Latrice
12Cowboyz 00Welcome 01Megenza 13croatia 00Zamboni 01Hannahg
1Michael 13mandala 01Arianna 01Inferno 01Olimpia 13samurai

Prepending with 3 Numbers:

000Welcome 003Kenneth 009Bhuvana 434Western 866Rathman
001Bhuvana 003welcome 010Bhuvana 444Chelsea 888Welcome
001sanyika 004Bhuvana 012Brownis 444Kriszta 888Zachary
002Bhuvana 005welcome 100Alissar 456Cowboys 888userpwd
003Aisling 006welcome 100IMedley 456Macbook 003Bhuvana
007welcome 429Wedding 456Markske
Prepending with 4 Numbers:

1236weather  1921Wedding  3029Jessica  4119Fairway
8224Hunterz  1315Marvick  1928Lorenzo  3042Sfiling
4122Wolfsun  8231Incubus  1324Booglet  2007Chaunce
2049Kelibia  4144Carroll  9234Account  1513Brandon
2015Wedding  2060belmont  4166Buffalo  9234Cheengr
1522Salinas  2017Jasmine  2075Jasmine  5199Eturkey
1526Katelyn  2021WestGe  2101Wedding  5210Ansarah
1800troyboy  2023Jillian

Prepend with 2 Numbers - Append 2 Numbers:

10Nico58  12cool12  56Moto97  20july83  55LOVE15
20Nori02  42csva07  01Polo86  30july95  85NIKE15
30PINK03  72cute92  46Quag17  12july99  45PAME15
40ango10  84baby12  0gamm12  50owen17  18amen02
50anna08  94bkjf41  60gary12  90paco12  48apoi89
70asis83  34cali14  30gfrc05  60paul17
Some password complexity rules require users to use a special character in their password. Users treat these characters differently than letters, and use them sparingly.

Appending Special Characters:

<table>
<thead>
<tr>
<th>thadeus!</th>
<th>antonio!</th>
<th>bubbles#</th>
<th>Californi@</th>
<th>CapeCoral@!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplyred!</td>
<td>FRANCE#</td>
<td>BonJovi@</td>
<td>dolphin,</td>
<td>Venice@!</td>
</tr>
<tr>
<td>Izabella!</td>
<td>GRANNY#</td>
<td>CITRIX@</td>
<td>cactus!@</td>
<td>change@!</td>
</tr>
<tr>
<td>NYGiants!</td>
<td>Student#</td>
<td>Computer@</td>
<td>cheese!@</td>
<td>Cowboy@@</td>
</tr>
<tr>
<td>changeme!</td>
<td>abcdefg#</td>
<td>clover!@</td>
<td>cheese@@</td>
<td>Dancing!</td>
</tr>
<tr>
<td>badgirl#</td>
<td>Austin!@</td>
<td>Dreamcatcher@</td>
<td>Lovemybabies@</td>
<td></td>
</tr>
</tbody>
</table>
Special Characters

Also, *prepend* a word with special character(s):

<table>
<thead>
<tr>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
<th>Word 4</th>
<th>Word 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>!!autumn</td>
<td>@@EMILY</td>
<td>@!pq10MZ</td>
<td>!1Clipper</td>
<td>!1q1q1q1q !!</td>
</tr>
<tr>
<td>deanna</td>
<td>!!Terry08</td>
<td>@!qp10MZ</td>
<td>!1Jamie5</td>
<td>!1qaz2wsx</td>
</tr>
<tr>
<td>##archie</td>
<td>!!abc123</td>
<td>@!summer05</td>
<td>!1Jamie5</td>
<td>!4London</td>
</tr>
<tr>
<td>@!andrew</td>
<td>!!alex85</td>
<td>!1London</td>
<td>!8Monday</td>
<td>@#Winter</td>
</tr>
<tr>
<td>!!die4me</td>
<td>!0Passwo</td>
<td>!1Monster</td>
<td>!9London</td>
<td>@@August</td>
</tr>
<tr>
<td>%@1234A</td>
<td>!0babies</td>
<td>!1Rebecca</td>
<td>!Firefox24</td>
<td>@@DAVID</td>
</tr>
<tr>
<td>!@Cancer</td>
<td>!1Bridge</td>
<td>!1Sunny</td>
<td>!Linux01</td>
<td>@#summer07</td>
</tr>
</tbody>
</table>
Special Characters

Additional "Specials" Patterns:
Append1_AddSpecialEverywhere:

Africa!1   AmyOct!1   Kar!dani1   XF!LES1   m!dnight1   Ahoney!1
Andyyu!1   john!deere1   abi!abi1   Alaya!1   AB!gail1
T@Y!OR1   b@byg!r1   Amanda!1   A!lison1   We!come1
S!LVER1   Amelia7!1   Ra!stlin1   Welcome!n1   S0lar!s1

Append 2010Special:

oct2010@   2Cute2010!   october2010!   Aaron2010!   Dublin2010!
pats2010!   March2010!   eagles2010!   Asia2010#   eagles2010$
Brooklyn2010@   eagles2010$   odessa2010$   Alyssa2010!
Homegirl2010$   Januarybaby2010!
Special Characters

Append4NumSpecial:

Abby1958$ Alex1005$ Andre7000! August2008$ Chris0707$
Abcd1234! Alex1109$ Athens2004$ August2008. Abcd1234$
Alex1209! April2007# August2010! Dece2010$ Abcd1234^$
Alex2008! August2008! August2010$ Alex0908$ Alex2010!
August2008# Canada2010!

Append6NumbersSpecial:

sl553015! uu124578! iggy104215! Fab240899! Greg731133#
sl553016! oct192010! baby112108! Feb020905! ss201011!
ee124578! Feb102010! tt124578! eric200509!
Feb190207! ty092906! fall051684! Fab240895!
Grace291133! Sherry123456! Summer200810!
AppendNumberNumberSpecialSpecial:

AUTUMN08$$ Andrew49$$ Baby33!! Cadillac44$$ January08$$
AUTUMN09$$ Aquaman12$$ Bold08$$ Dolfan00$$ Abcd16$$
AteO00.. Saints01$$ Emma08!! Alex29$$ August09$$
Cadillac33@@ Football33@@ Alex33$$ Baby09!! Cadillac33##
Henry23$$

AppendSpecial4num:

Amy!2006 Aug#2010 Baby@0303 1nM@7352
Anala@2002 Aug@1826 Baby@1628 Anand@1980
Aug@2000 Dallas@2010 September@2005 11vaca@2006
Andrew@1 Aug@2004 mn@2008 11Fire@0601
alexandria!2010 September#2010
Users also love finger patterns:

NHY%^tgb  qwe~123  !23qweasd !QWERTY  NHY^5tgb
!1234qwe  !@$QWE  ASDFqwer qwertyqw  !123qwer
!@#123qwe (123qwe)  ASDFqwer##  1qwe!QWE
%1QWertyuiop

1) john --external:keyboard (works, but is not perfect)
2) make your own (Or use KoreLogic’s)

Sometimes during corporate training classes, they will tell users to use this method. Take advantage of this fact!
Internet related (These work REALLY well on Internet Apps):
0900.com   1723.com  YaL@agf.com  KOM.NET  aafp.org
1395.com   17jm.com  gabiEiel.com  capi.net  woop.org
1828.com   1995.COM  @cox.net  ARMY.ORG

Dev/Prod/Test/UAT Related:
Prod!111 prod@123 TEST-CO test!ng  webtest
Prod!121 UAT$2109 TEST-DO test-1234 Prod!131
Uat$1234 Test@1109 test.123 silktest Prod=332
Uat$2010 Test@123 testftp preprod1 Prod=666
TEST-CA Test@cct CFPTTEST preprod! prod4321
So you've used all these rules to crack some passwords, what else can you do?

Use your cracked password as a dictionary!

```
# john -show --format:nt pwdump.txt | cut -d: -f 2 | sort -u > cracked.dic
or
# cut -d: -f2 john.pot | sort -u > cracked.dic
```

This is especially useful if you have:
1) Hashes from multiple systems
2) Password history files - (or hashes from previous months/years).
3) Large amounts of users

OWASP: Use the 'rockyou' list against your hashes!

Reuse all the "rules" using the new cracked.dic as the wordlist.
Also..... (next slide)
Search/Replace Rules:
John's default john.conf has some of these. We have improved them.

Numbers -> Specials:
Austin1 -> Austin!
Testing222222 -> Testing222@@@
Dallas9 -> Dallas(

Replace Numbers -> Other Numbers:
$Austin01  $Austin05  $Austin09  Austin!01  Austin!06
$Austin02  $Austin06  @Austin73  Austin!02  Austin!07
$Austin03  $Austin07  @Austin74  Austin!04  Austin!09
$Austin04  $Austin08  @Austin76  Austin!05
## Replace Letters -> Other Letters

<table>
<thead>
<tr>
<th>Original</th>
<th>New Letter</th>
<th>Original</th>
<th>New Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>austin12</td>
<td>bustin12</td>
<td>austin21</td>
<td>dustin21</td>
</tr>
<tr>
<td>Austin37</td>
<td>Austin37</td>
<td>AUSTIN3</td>
<td>AUSTIN3</td>
</tr>
<tr>
<td>BOSTON1</td>
<td>DOSTON1</td>
<td>BOSTON1</td>
<td>FOSTON1</td>
</tr>
<tr>
<td>Austin11</td>
<td>Bustin11</td>
<td>Austin1$</td>
<td>Rustin1$</td>
</tr>
<tr>
<td>Austin08</td>
<td>Tustin08</td>
<td>Austin16.</td>
<td>Vustin16.</td>
</tr>
<tr>
<td>Password1</td>
<td>Cassword1</td>
<td>Password#1</td>
<td>Wassword#1</td>
</tr>
<tr>
<td>password</td>
<td>password</td>
<td>password</td>
<td>casassword</td>
</tr>
</tbody>
</table>

Users will choose a password such as “Vegas2010” but when they are forced to change it – they will just change a single letter.

**Example:** Wegas2010 Vegaz2010
Longest Passwords Cracked by KoreLogic:

- Abcdefghijklmnopqrstuvwxyz 12345678901234567890
- my chemical romance  representative2118
- Robbiewilliams1234 happybirthday2005!
- Prideandprejudice communication2000
- Astralprojection cheesecake041004
- Sleadadministrator Januarybaby2010!
- Cheesecake041004 specialized7777
- Rasheedwallace7 rememberthename
- Waterville2008$ smartyjones2008

Do you see any pattern here that is super strong?

Do you see any pattern here that we haven’t talked about?
WordLists

Wordlists:
- Having a variety of wordlists is required.
- KoreLogic has the most luck with the following lists:

Seasons - Months - Years - First Names - Last Names - Cities - States - Regions - Countries - "RockYou" List - Regions of India/China/USA - Religious references (books of the Bible, lists of Gods, etc) - keyboard combinations - 4 letter words - 5 letter words - 6 letter words - 7 letter words - Sports Teams - Colleges - Client specific words - Dates - Numbers - Common wordlists – Facebook Names List (‘fbnames’) etc. etc.

Even without any new rules - these types of dictionaries are more likely to crack more passwords than the defaults.

KoreLogic has released a large set of wordlists that can be used with any password cracker you wish. http://contest.korelogic.com/
For a small amount of money (< $4000) you can build an amazing password cracking system using “off the shelf” video cards usually reserved for high-end gaming.

KoreLogic’s system has 4 GTX 480 cards (Approx $500 each). Capable of cracking 6220 Million hashes a second

This is the “future” of high-end password cracking. (Note: New version of IE will use GPU processing power to render pages)

Formats Supported:
- NTLM (Windows / Active Directory)
- SHA/SSHA (LDAP Directory Format)
- MD5 (various variations)
- DES (Used by UNIX Systems).
Brute force is _much_ easier with GPU systems (and much much faster).

OclHashCat ("best" GPU password cracking software) includes many tools for better/smarter/faster password cracking.

KoreLogic can identify 100% of all 8 character (or less) from an Active Directory (NTLM) in days using GPU cards. It used to takes weeks.

(What does your security policy have for its minimum length required of passwords? 8 is no longer enough, make it 9 or 10)

The winning team at our DEFCON contest, had 11 high-end GPU cards. This number is up to at least 16 now. Including one system with 6 high-end (liquid cooled) GPU cards in it.
Running the default rules/wordlists/methods of whatever password cracking tool you use is OK. It’s not going to hurt.

Using better rules/wordlists (based on actual password data) is better and more likely to crack additional passwords.

Why not improve the rules/wordlists/methods to crack better passwords?

Password complexity rules do not make users choose "stronger" passwords.

Complexity rules encourages users to use patterns/tricks to remember the stronger passes. Abuse these patterns/tricks.

Password rotation _introduces_ new password patterns.
Web applications do not even enforce password complexity! Why not?

All this “logic” is overkill when cracking “Internet” passwords because users choose horrible passwords.

Web Applications should start enforcing password complexity now and catch up with Operating Systems.

2 Factor in being pushed now – Google/PayPal/WoW. Get on it!

If the password hash format your application uses is not support by John/HashCat/OclHashCat – how are people going to crack them?

Please use 'bcrypt' to hash your passwords!!
1) User Awareness. Make your users aware that you know what they are doing. It is better to train them. Tell them to stop using Month/Year/Seasons in the passwords.

•Tell them what makes a better password. (Length, Randomness, Special characters in the middle of the password).

2) Technology: Make your password complexity requirements aware of these patterns. Users should not be allowed (by both Policy and Technology) from using these patterns/wordlists.

3) Routinely Audit passwords.
   This used to be a "no no" in corporate environments.
   Now, its a 'yes yes'. Ask for help.

   Learn what patterns your users are using.
1) User Awareness. Make your users aware that you know what they are doing. Train them to choose better passwords:
   - Google Technique – JavaScript

2) 2-Factor (PayPal / Google / WoW)

3) Do not allow users to use CERTAIN passwords (Twitter Method)
Ideas for improvement:
Make a "free lunch" contest for all users whose password do not crack with 24 hours.

   In the announcement of the contest, teach/train users about methods for creating complex “uncrackable” password patterns.

Example: "Chunk Norris is gonna beat! you up" → cnIGB!Yup
Example: “Oh, I see you ate one too” → oh,lcU812
Example: “Something under the bed is drooling!” → "5u+B!drl"
Example: “NeverGonnaGiveYouUp” → nvrGONNA!!givu.
Example: “Wow, I was impressed with KoreLogic” ->
“\W0w,KoreRules” ;)

This combination of incentive, testing, prevention, auditing, and training will help your organization create stronger passwords.
Previous Events:

**Defcon 2010** - KoreLogic sponsored a password cracking contest. 54,000 hashes were released, and teams had 48 hours to crack as many as possible. $1,000 in prizes.


Upcoming Events:

**DEFCON 2011** – ‘Crack Me If You Can’ Contest:

**DerbyCon 2011** – New Security Conference in Louisville, KY. KoreLogic will be present. Possibly running a mini-contest for all password crackers.
Read:
http://contest.korelogic.com/
for all wordlists/tips/tricks/rules/examples/etc


Q &A:
Rick Redman – KoreLogic

rredman@korelogic.com
http://www.korelogic.com/