Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

# Detecting Spam with Genetic Regular Expressions

Copyright SANS Institute
Author Retains Full Rights

AD

Beating the IPS?

STONESOFT
Can't be Beat

Learn
More

© SANS Institute 2007, Author retains full rights.

# Detecting Spam with Genetic Regular Expressions

*GCIA Gold Certification*

Author: Eric Conrad, eric@ericconrad.com

Adviser: Johannes Ullrich

Accepted:



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

# Outline

1. Abstract ..... 3
2. Introduction ..... 4
3. An Approach to Automatically Detect Spam with Genetic Regular Expressions ..... 11
4. Genetic Regular Expressions ..... 18
5. Proof-of-Concept Script Results ..... 29
6. Conclusion ..... 43
7. Further study ..... 44
8. Appendices ..... 46
9. References ..... 66

© SANS Institute 2007, Author retains full rights.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

## 1. Abstract

Regular Expressions ("regex" for short) are strings used to detect patterns in data. They are often used to detect and block various forms of malware, including spam and network-based attacks.

This paper describes an approach for detecting spam with automatically-generated regular expressions (where regexes are generated according to simple logic), followed by a 'genetic' approach (where regexes are generated, and then 'evolve' to the final solution via a genetic algorithm).

This approach was inspired by the author's experience running inbound internet SMTP gateways for a 12,000 employee company. As of the fall of 2007, the gateways blocked 150,000 spam per business day.

© SANS Institute 2007, Author retains full rights.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

## 2. Introduction

### Problem Domain

The 'problem domain' (specific challenge) addressed by this paper is detecting spam emails via genetic regular expressions, focusing on 'Advanced Fee' scams. These scams include lottery scams (where the target is told he/she has won a lottery, and is asked to submit a fee to receive payment), and '419' scams.

A 419 scam is a scam where an alleged foreign government official requests an upfront payment in exchange for releasing large sums of money. It is named after the section of Nigerian legal code violated by the scam[1]".

The FBI's article titled 'Common Fraud Schemes' describes the scam:

> *Payment of taxes, bribes to government officials, and legal fees are often described in great detail with the promise that all expenses will be reimbursed as soon as the funds are spirited out of Nigeria. In actuality, the millions of dollars do not exist*

[1] According the International Centre for Nigerian Law : 'Any person who by any false pretence, and with intent to defraud, obtains from any other person anything capable of being stolen, or induces any other person to deliver to any person anything capable of being stolen, is guilty of a felony.' URL: http://www.nigeria-law.org/Criminal%20Code%20Act-Part%20VI%20%20to%20the%20end.htm



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

*and the victim eventually ends up with nothing but loss. Once the victim stops sending money, the perpetrators have been known to use the personal information and checks that they received to impersonate the victim, draining bank accounts and credit card balances until the victim's assets are taken in their entirety. While such an invitation impresses most law-abiding citizens as a laughable hoax, millions of dollars in losses are caused by these schemes annually. Some victims have been lured to Nigeria, where they have been imprisoned against their will, in addition to losing large sums of money. The Nigerian government is not sympathetic to victims of these schemes, since the victim actually conspires to remove funds from Nigeria in a manner that is contrary to Nigerian law. The schemes themselves violate section 419 of the Nigerian criminal code, hence the label "419 fraud*."[2]

The scam originated in Nigeria, but is now used around the world, describing various government and private sector officials with alleged access to large sums of money.

In the fall of 2007, the most spam complaints received at my company of 12,000 employees involved 419 scams. The inbound internet mail relays run Postfix, SpamAssassin, and antivirus software, and do an excellent overall job of blocking most types of spam.

[2] URL: http://www.fbi.gov/majcases/fraud/fraudschemes.htm



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Here's an example 419 scam email received 'in the wild':

```
To:
Subject: Hello
From: "villaran nenita" <villaran1976_n@citromail.hu>
Date: Thu, 13 Sep 2007 12:16:56 CEST

Hello Dear,

My name is nenita villaran,The wife of Mr.Panfilo Nenita a senator during p=
resident joseph Estrada regime in Philippine.who is receently killed in phi=
lippine.
During my husband's regime as a senator,I realized some reasonable amount o=
f money from various deals that I successfully executed and my buisness in =
the united state of which the Government has block most of my account in th=
e bank of america,trying to leave me
with nothing.

well before my late husband was killed,I secretly put in a box the sum of $=
30,000,000 million USD (Thirty million United states dollars) and deposit i=
t in a security company abroad.
I am contacting you because I want you to help me in securing the money for=
 the future of my children since the government now monitor all my movement=
 .
I hope to trust you as who will not sit on this money when you claim it.i w=
ill give you 15% of the total money for your assistance.if you are willing =
to help me as soon as possible

Best regards=20

Villaran Nenita[3]
```

The challenge in filtering these types of emails with software such as SpamAssassin is that they are comprised mainly of English text, designed to look somewhat like a business letter. They may be sent from any domain (no forging required), and do not typically contain a trackable URI, which may be used to identify spam via 'URI Blacklists.'[4]

This specific spam was detected by these SpamAssassin rules (among others):

```
__FRAUD_JYG ======> got hit: "give you 15% of the total"
```

[3] Personal email communication, September 17, 2007
[4] One example is the URI BL. URL: http://www.uribl.com/about.shtml



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
__FRAUD_NRG ======> got hit: "I am contacting you"
__FRAUD_LTX ======> got hit: "million USD"
__FRAUD_IRJ ======> got hit: "security company"
```
[5]

Here's a 419 scam email that was not blocked by SpamAssassin:

```
From: Fund Manager
Sent: Friday, September 07, 2007 9:46 AM
To: undisclosed-recipients
Subject: Mail..

Re: Confidential Deal:

Greetings,

 I am a Fund Manager with Fidelity Investment UK and I handle all our Investor's Capital
Project Funds that enables me to divert 1.2% Investors Excess Return Capital Funds to our
Magellan Trust Funds Account whereby anyone can be presented to claim the funds. On this
note, the total sum of US$34.5M has been diverted representing the 1.2% Excess Return
Capital Funds from the Investor Capital Project Funds for 2006/2007.I need a reliable and
trustworthy person that can work this deal out with me so that we can claim the funds as
mentioned above.

There is no risk attached and the funds in question can never be dictated or traced. Our
sharing ratio is 50:50.If you are interested,please send your direct telephone numbers
for discussion of this deal in further details. I am counting on your sense of
confidentiality,as it is my desire that you keep this business to yourself.

Sincerely.

Mr.Carlos Moreno.
```
[6]

It triggered these SpamAssassin rules:

```
__FRAUD_YPO ======> got hit: "the total sum"
__FRAUD_IOU ======> got hit: "no risk"
```

Although it triggered some rules, the overall SpamAssassin score was under the site's quarantine threshold, so the email was passed onto the end user (who later complained to the internet mail team).

---

[5] Output generated with 'spamassassin –D', showing lines with "FRAUD'. Cleaned up for formatting purposes
[6] Personal email communication, September 3, 2007



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

## Regular Expressions

Regular expressions are a powerful mechanism for matching text. A simple regular expression matches literal text.[7] Regexes may also contain advanced features such as metacharacters, character classes, and grouping (among others).[8]

Metacharacters are concise commands which perform a function, such as 'match the start of line' ("^"). Character classes allow a range of characters, such as [a-z] for lowercase characters. Grouping allows a choice of words, like "(FEE|FIE|FOE|FUM)".

This example uses all three types:

```
/^You are in a maze of twist(y|ing) little passages[, .]/
```

This regex matches lines beginning with "You are in a maze of …", allows the choice of 'twisty' or 'twisting', and matches a space, period, or comma after 'passages'.

Perl Compatible Regular Expressions (PCRE)[9] are a powerful regular expression engine derived from the Perl programming language.[10] Many software packages use PCREs,

---

[7] `/This is a literal match/`
[8] A good resource for learning about regular expressions is *Mastering Regular Expressions by* Jeffrey Friedl.. O'Reilly & Associates, January 1997. ISBN 0-596-00289-0.
[9] PCRE - Perl Compatible Regular Expressions. URL: http://www.pcre.org/
[10] Information about Perl may be found at The Perl Directory at Perl.org . URL:; http://www.perl.org



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

including SpamAssassin, Snort, Apache, Postfix, and many others. This paper uses PCREs in all examples and code.

## Leveraging Regular Expressions to Block Malware

Regular expressions are harnessed in many types of software, including intrusion detection systems and spam-blocking software.

Here's an example of a Snort rule including a PCRE:[11]

```
alert tcp $EXTERNAL_NET 22 -> $HOME_NET any (msg:"EXPLOIT SSH server
banner overflow"; flow:established,from_server; content:"SSH-"; nocase;
isdataat:200,relative; pcre:"/^SSH-\s[^\n]{200}/ism";
reference:bugtraq,5287; reference:cve,2002-1059; classtype:misc-attack;
sid:1838; rev:8;)
```

A simplified description of this rule is that it will match on traffic with a TCP source port of 22 (the SSH daemon port), matching the PCRE "/^SSH-\s[^\n]{200}/ism".

That means: match the literal string "SSH-" at the beginning of the line, followed by at least 200 characters that are not newlines. The search is case insensitive ('i'), will match across multiple lines ("m"), and '.' ('dot' metacharacter which matches any character) will also match a newline ("s").

[11] Official Snort 2.4 rules. URL: http://www.snort.org/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.4.tar.gz, /rules/exploit.rules



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Here is an example of a SpamAssassin rule containing a regex:

```
Subject =~ /(Your Lucky Day|(Attention:|ONLINE) WINNER)/i
```
[12]

That matches a Subject which includes any of the following:

- `Your Lucky Day`
- `Attention: WINNER`
- `ONLINE WINNER`
- `OnLiNe wInNeR`

That last example matches because the search is case sensitive.

---

[12] SpamAssassin 3.2.3. URL: http://mirror.fslutd.org/apache/spamassassin/source/Mail-SpamAssassin-3.2.3.tar.gz, Mail-SpamAssassin-3.2.3/rules/72_active.cf

© SANS Institute 2007, Author retains full rights.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

# 3. An Approach to Automatically Detect Spam with Genetic Regular Expressions

## Bayesian Filtering with Spam and Ham

While regular expressions are typically written by humans, there are classes of software that use statistical methods to automatically detect spam. One such approach is Bayesian filtering, named after Thomas Bayes, an English clergyman who devised a number of probability and statistical methods, including "a simple mathematical formula used for calculating conditional probabilities."[13]

Paul Gram described Bayesian filtering to identify spam in his paper 'A Plan for Spam.'[14] He described using a 'corpus' of 'spam' and 'ham', human-selected groups of spam and non-spam., respectively He then used Bayesian filtering techniques to automatically assign a mathematical probability that certain 'tokens' (words in the email) were indications of spam.

We will use the spam and ham corpus approach to identify the fitness of genetic

[13] Joyce, J. (2007). Bayes' Theorem, *The Stanford Encyclopedia of Philosophy (Summer 2007 Edition), Edward N. Zalta (ed.).* URL: http://plato.stanford.edu/archives/sum2007/entries/bayes-theorem/
[14] URL: http://www.paulgraham.com/spam.html



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

regular expressions.

## Automatic Regular Expressions

Regular expressions leveraged to block spam or malware may be automatically generated according to the following logic:

1. Create 2 sets of emails called 'spam' and 'ham'
2. Break spam into tokens
3. Generate a large number of regular expressions based on the spam tokens
4. Delete regex that match any ham
5. Delete regex that match only 1 spam
6. Rank the remaining regex, sorted from highest number of matching spam

The regular expressions are generated using the following algorithm, applied in order (first match wins):

- If its a number, replace with \d+ (1 or more digits)
- If its capital hexadecimal, replace with [A-F0-9]+ (1 or more capital hex digits)
- If its lowercase hexadecimal, replace with [a-f0-9]+ (1 or more lowercase hex digits)
- If its a common TLD, replace with (com|net|org|edu|biz|info|us)
- If its a lowercase word, replace with [a-z]+ (one or more lowercase letters)
- If its an uppercase word, replace with [A-Z]+ (one or more uppercase letters)
- If its an abbreviated day of the week, replace with (Mon|Tue|Wed|Thu|Fri|Sat|Sun)



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

- `If its an abbreviated month, replace with (Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)`
- `If its a word with the first letter capitalized, replace with [A-Z][a-z]+ (uppercase letter, followed by one or more lowercase letters).`
- `Else: keep the token as a literal regex.`

Please see Appendix A for proof-of-concept Perl source code of the above logic.

These rules were run against a small corpus of spam (339) and ham (410) from personal email. Some spam were previously tagged with "***SPAM***" in the subject line (by SpamAssassin); the tag was left in to determine if any regexes would emerge that matched that tag.

The following regular expressions were generated, ranked in order of spam matched:

```
Count regex
-----------------------------------------------------------------
22 /^From: "[A-Z][a-z]+ [A-Z][a-z]+" <[a-z]+@[a-z]+\.[a-z]+\.[a-z]+>/
27 /^Subject: [a-z]+ [a-z]+ [a-z]+/
32 /^Subject: [a-z]+ [a-z]+/
33 /  +[a-z]+="----=_NextPart_\d+_\d+_[A-F0-9]{8}\.[A-F0-9]{8}"/
37 /^Subject: \*\*\*[A-Z]+\*\*\* [A-Z][a-z]+ [a-z]+ [a-z]+/
43 /^Subject: \*\*\*[A-Z]+\*\*\* [A-Z][a-z]+ [a-z]+/
52 /  +[a-z]+="----=_NextPart_\d+_[A-F0-9]{4}_[A-F0-9]{8}\.[A-F0-
9]{8}"/
57 /^Subject: \*\*\*[A-Z]+\*\*\* [A-Z][a-z]+/
61 /^From: "[A-Z][a-z]+ [A-Z][a-z]+" <[a-z]+@[a-
z]+\.(com|net|org|edu|biz|info|us)>/
88 /^From: "[A-Z][a-z]+ [A-Z][a-z]+" <[a-z]+@[a-z]+\.[a-z]+>/
```

"***[A-Z]+***" is a regex that matches "***SPAM***": three asterisks ("*"), followed



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

by an uppercase word (1 or more of the capital letters A-Z), followed by three asterisks.[15]

Here are some matches:

```
Subject: ***SPAM*** And who pronunciation
Subject: ***SPAM*** Are lacamp he estherville
Subject: ***SPAM*** Are well coahoma
Subject: ***SPAM*** As she kennedyville
```

This tag (inserted by SpamAssassin) was left in the spam corpus as a test, to see if regex would be generated which match it.

The regex:

```
/  +[a-z]+="----=_NextPart_\d+_[A-F0-9]{4}_[A-F0-9]{8}\.[A-
F0-9]{8}"/
```

…matches a MIME (Multipurpose Internet Mail Extension) attachment headers:

```
boundary="----=_NextPart_000_000F_01C78661.D9DA98C0"
boundary="----=_NextPart_000_000F_01C78818.3AAE1BB0"
boundary="----=_NextPart_000_000F_01C788D2.D850C470"
boundary="----=_NextPart_000_000F_01C78EAE.D04A6A60"
```

They appear to be legitimate MIME boundaries to the naked eye. The regex ‘  +’ means ‘match two or more spaces.’ The ham corpus contained similar headers, but those began with a leading ‘tab’ character (\t). Some mail in the spam corpus replaced the tab with

[15] Note that‘*’ is a metacharacter (which means ‘match zero or more of the previous character’) which must be



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

8 leading spaces, which was a reliable indicator of spam within the corpus.

The automatic method of generating regexes appears to be effective.

## Genetic Algorithms

A Genetic algorithm (GA) is an algorithm that is automatically generated, and then 'bred' through multiple generations to improve it via Darwinian principles:

> *Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance.*[16]

'Genetic regular expressions' may be developed by combining genetic algorithms, the

---

escaped with a backslash in order to be treated as a literal asterisk.

[16] Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

spam and ham corpus approach used in Bayesian filtering, and automatic regular expressions described in the previous section. The goal is to solve the complex problem of matching 419 spam text. As genetic algorithm pioneer John Holland said "Computer programs that 'evolve' in ways that resemble natural selection can solve complex problems even their creators do not fully understand."[17]

## Key GA Terms

Genetic algorithms use chromosomes made up of individual genes. Chromosomes are designed to perform a specific function, such as sorting numbers, playing tic-tac-toe, or, in this case, matching spam text.

Chromosomes are scored according to a fitness function. Higher-scoring chromosomes are more likely to survive and are able to breed future generations. Breeding includes mating with other chromosomes and exchanging genes via a 'crossover' function. Stronger chromosomes are more likely to be chosen to breed. One way to select parents is 'Roulette Wheel' selection, described below.

Mutation may occur, where a random change is made to the chromosome. Many

[17] Holland, John H. "Genetic Algorithms," *Scientific American,* July 1992, URL:



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

mutations may be damaging, but some could improve the 'health' of the chromosome.

A typical genetic algorithm uses the following logic:[18]

- Create a large number of chromosomes
- Loop until complete:
  - Have the chromosomes perform a specific function
  - Apply a fitness score to each chromosome
  - Identify the strongest chromosomes and allow them to survive to the next generation
  - Breed new chromosomes by selecting genes from 2 parents
  - Mutate some genes

The loop ends at a fixed time (100 generations, for example), or when a certain condition is met, such as laying a perfect game of tic-tac-toe, or sorting integers in a specific number of steps.

---

http://www.econ.iastate.edu/tesfatsi/holland.GAIntro.htm

[18] For more background on practical genetic algorithms, including detailed pseudocode, see Dracopoulos, Dimitris C. *2AIT608 - Machine Learning Genetic Algorithms*. URL: http://users.wmin.ac.uk/~dracopd/DOCUM/courses/2ait608/genetic_algorithms_lecture_notes.pdf



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

## 4. Genetic Regular Expressions

A regular expression chromosome is a full regular expression. Genes within the chromosome are literals, strings, metacharacters, and character classes (or groups thereof).

Here is an example regular expression chromosome:

```
Subject: \*\*\*[A-Z]+\*\*\* [A-Z][a-z]+ [a-z]+/
```

Here are the individual genes:

1. `Subject:`
2. `\*\*\*`
3. `[A-Z]+`
4. `\*\*\*`
5. `[A-Z][a-z]+`
6. `[a-z]+`

The following process is used to evolve genetic regular expressions:

1. Generate a population of regex chromosomes based on the automatic method described above.
2. Match them against the spam and ham corpora[19]
3. Assign a fitness score
4. Choose the fittest chromosomes to survive to the next generation (the rest 'die')
5. Crossover genes between some parents to produce children
6. Mutate genes in some children

[19] The plural of 'corpus' is 'corpora,' although 'corpuses' may also be used. See: http://wiki.apache.org/spamassassin/PluralOfCorpus?highlight=%28ham%29%7C%28corpus%29



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
7. Repeat process for X generations
```

### Genetic Regular Expressions Psuedocode

The full Perl source code to create genetic regular expressions is included in Appendix B. Here is the same logic, in pseudocode:

```
1. Choose the number of generations (we'll use 10)
2. Read the spam and ham corpora
3. Randomly shuffle the lines of spam
4. Divide spam corpus into 10 slices
5. Loop until 10th generation:
     a. Generate chromosomes based on the current slice of
        spam using the 'automatic' formula
     b. Score chromosomes
     c. Print results for the current generation
     d. Keep the fittest 3rd
     e. Breed survivors
         i.   2 survivors breed via a crossover function to
              create a child
        ii.   Use 'Roulette Wheel' selection top choose the
              2nd parent
     f. Mutate some of the children by randomly deleting some
        genes
     g. Move to next slice of spam
6. Print Final results
```

### Sources of ham and spam

The ham used in this paper is based on the SpamAssassin 'easy ham' corpus,



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

downloaded from the SpamAssassin Public corpus.[20] The spam corpus was collected from my company's inbound internet relays from September 12th-14th, 2007.

The SpamAssassin public ham corpus was chosen due to its high quality, and because it was pre-cleaned of any sensitive information (such as names, IP addresses, etc). Current 'advanced fee' spam was chosen because it is required for addressing the problem domain of blocking 'advanced fee' scam emails currently bypassing spam filters. Also, the spam bodies contain no sensitive information.

The spam and ham corpora used in this paper are publicly available, along with the POC code. Please see the appendices for more information,

Both corpora were formatted according to the following logic:

```
1. Remove the headers
2. Remove all lines not consisting solely of the following
   characters:
      a. Alphanumeric
      b. Whitespace
      c. Comma
      d. Period
      e. Underscore
3. 'sort' and 'unique' the lines[21]
```

The goal is to isolate lines containing human-written text, and skip headers, MIME

[20] URL: http://spamassassin.apache.org/publiccorpus/20030228_easy_ham.tar.bz2



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

information, etc.

## Automatic chromosomes

The proof-of-concept scripts seeds each round with chromosomes based on the spam corpus. The following logic is used in order to introduce randomness in the seeding:

For each prospective gene, randomly choose one of the following:

1. Leave it as a literal regex
2. Change it to a regular expression, using the 'Automatic Regular Expression' rules described previously
3. Change it to '.*', meaning 'match zero or more of any character'

Convert any whitespace to "\s+"[22]. Then combine any repeated ".*"[23]

During the example run below, this line of spam:

```
me to Register it as  Consignment with GLOBAL TRUST COURIER
COMPANY.
```

…was converted to this chromosome, using the above logic:

```
/^[a-z]+\s+to\s+[A-Z][a-z]+\s+[a-
z]+\s+as\s+Consignment\s+[a-z]+\s+.*[A-Z]+\s+COURIER\s+/
```

---

[21] On Unix, 'sort –u <unsortedspam.txt >sortedspam.txt'
[22] 'match 1 or more whitespace characters'
[23] '.*.*' is the same as '.*', but less efficient.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

## Fitness Function

Chromosomes that match any ham 'die' (are ignored by the program). The surviving chromosomes are then scored by the fitness function.

The fitness function has 2 components: number of spam lines matched, and length of the chromosome. Short chromosomes that match lots of spam receive high scores; long chromosomes that match little spam receive low scores.

Remaining chromosomes are scored according to this fitness function:

```
(Number of lines of spam matched) *
(1+((200-<length of chromosome>)/200));
```

In other words, any chromosome under 200 characters long will receive a fitness bonus; chromosomes over 200 characters receive a penalty.

A 100 character chromosome matching 5 lines will receive a fitness 'bonus' due to it's comparatively short length, and receive a score of 7.5.[24] A 300 character chromosome matching 5 lines of spam will be 'punished' for its long length, and receive a score of 2.5.[25]

---

[24] 5 * (1+ (200-100)/200) == 5 * (1+ 100/200) == 5 * (1+ .5) == 5 * 1.5 == 7.5
[25] 5 * (1+ (200-300)/200) == 5 * (1+ -100/200) == 5 * (1 - .5) == 5 * .5 == 2.5



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Without the length 'bonus,' chromosomes will grow would longer and longer each generation (simply adding other winning genes via breeding). The fitness function is designed to reward shorter/more efficient chromosomes, and keep the length manageable.

### Roulette Wheel Selection

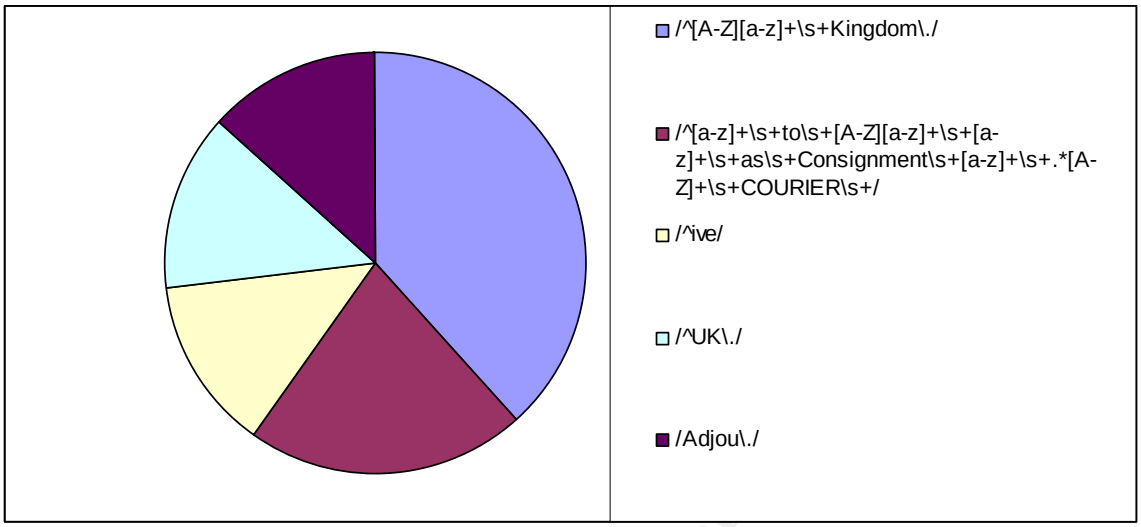
Roulette Wheel selection is a method for choosing chromosomes from a breeding pool. Higher-scoring chromosomes are more likely to be chosen than lower-scoring: a chromosome with a score of 3 is three times more likely to be chosen than a chromosome scoring 1.

> *An imaginary roulette wheel is constructed with a segment for each individual in the population... the size of the segment is based on the aptness of the particular individual. A fit individual will occupy a larger slice of the roulette wheel than a weaker one.*[26]

Here are the 5 highest-scoring chromosomes from the 1st generation of our results (see below). `'/^[A-Z][a-z]+\s+Kingdom\./'` scored 5.64, compared with `'/Adjou\./',` which scored 1.96. The latter has a proportionally larger 'slot' in the roulette wheel, and is therefore more likely to be chosen for breeding.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.



[27]

## Crossover Function

Our crossover function creates 1 child from 2 parents, sharing genes from both. Another approach is to create 2 children from 2 parents, each sharing a subset of the parent's genes.

Each survivor from the previous generation creates a child chromosome. A 2nd parent is chosen from the breeding pool via Roulette Wheel selection. We have 2 fundamental crossover options: breaking down and combining all genes via an 'or' function (the 'Or

---

[26] Dalton, John. *Newcastle Engineering Design Centre, January 2007: Genetic Algorithms (GAs).* URL: http://www.edc.ncl.ac.uk/highlight/rhjanuary2007.php

[27] Example simplified with 5 chromosomes for clarity. A 'full' roulette wheel would include all surviving chromosomes in its breeding pool.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Method'), and concatenating 2 parent's chromosomes (the 'Cat Method').

### The 'Or Method'

An example of the 'Or Method' takes 2 parent chromosomes, such as '(FOO|BAR)' and '(BAR|BAZ)', breaks them down to their unique genes, and 'ors' them together to create a child called '(FOO|BAR|BAZ)'.

Here's a specific example, taken from the generation 2 output, below. Here are 2 parents:

```
/(?:^DR\.\s+[A-Z][a-z]+\s+[A-Z][a-z]+|^[A-Z][a-z]+\s+Kingdom\.)/
/(?:^London,England|^[A-Z][a-z]+\s+Kingdom\.)/
```

The 'Or Method' breaks them down to their unique genes:

- `^London,England`
- `^[A-Z][a-z]+\s+`
- `Kingdom\.`
- `^DR\.\s+`
- `[A-Z][a-z]+`

It then creates a child chromosome by combining them in 'OR' ('|') fashion:

```
/(?:^London,England|^[A-Z][a-z]+\s+|Kingdom\.|^DR\.\s+|[A-Z][a-
z]+)/
```

Barring mutation (see below), this child is unlikely to survive. It includes the gene '`[A-`



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

`Z][a-z]+',` which will match any capitalized word. The nature of crossover is both weaker stronger children will be created, and the strongest will survive.

### The Cat Method

The 'Cat Method' takes 2 parent chromosomes and concatenates them together, creating a longer chromosome. Take these 2 parents:

```
/(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance
)/

/YOUR\s+[A-Z]+\s+[A-Z]+/
```

Simply concatenate them, taking (/Chromosome1/) and (/Chromsosme2/), creating the child (/Chromosome1|Chromsosme2/):

```
/(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance
|YOUR\s+[A-Z]+\s+[A-Z]+)/
```

### Mutation

The mutation function takes a child chromosome, and changes it in a randomized way. The simplest method is deleting a random amount of genes.

Our mutation function takes a child gene, such as the child from the last section:

```
/(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance
|YOUR\s+[A-Z]+\s+[A-Z]+)/
```

It then randomly deletes some genes:



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
/(?:LOTTERY\s+WINNING\s+|^deposit|the\s+finance|YOUR\s+[A-
Z]+\s+[A-Z]+)/
```

Since shorter chromosomes are considered more 'fit' than equivalent longer chromosomes, mutation may help the fitness of a child. Also, mutation may remove some harmful genes, such as '[A-Z][a-z]+'.

As with Crossover, some mutations are harmful to children, but some may be helpful to the fitness of the child.

### A word on efficiency

Due to the large amount of regular expression checks required in this process, the checks should be as efficient as possible. This regular expression contains grouping:

```
/Read the Jargon File to learn about the metasyntactic
variable (Foo|Bar|Baz)/
```

In addition to grouping 'Foo', 'Bar' and 'Baz', it also *captures* the resulting match. To check to see what was actually matched in Perl, refer to variable "$1":

```
print "The previous regex matched $1\n";
```

Capturing takes extra time and memory, so it is wasteful if the results of the capturing are not needed. Insert "?:" after the opening parentheses In order to avoid capturing (and make the regex more efficient):

```
/This regex groups but does not capture: (?:Foo|Bar|Baz)/
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

The code used in this paper will use efficient constructs such as '?:' Where appropriate. The proof-of-concept script uses other efficiency improvements, such as pre-compiling the regexes. See the POC code in Appendix B for more details.

© SANS Institute 2007, Author retains full rights.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

## 5. Proof-of-Concept Script Results

Here are the results of 'genregex.pl' (see Appendix B for full source code), running for 10 generations. The top 5 chromosomes are shown for each generation.

Note: these results are summarized and formatted into tables for clarity. The full text output is shown in Appendix C.

### Generation 1

| Score | Chromosome |
|---|---|
| 5.64 | /^[A-Z][a-z]+\s+Kingdom\./ |
| 3.13 | /^[a-z]+\s+to\s+[A-Z][a-z]+\s+[a-z]+\s+as\s+Consignment\s+[a-z]+\s+.*[A-Z]+\s+COURIER\s+/ |
| 1.98 | /^ive/ |
| 1.98 | /^UK\./ |
| 1.96 | /Adjou\./ |

The first generation contains automatic chromosomes, with no breeding or mutation (yet). The fittest chromosome, '/^[A-Z][a-z]+\s+Kingdom\./[28],' matches spam lines beginning with "United Kingdom." It matches these three lines in the spam corpus:

```
# perl -e 'while(<>){print if (/^[A-Z][a-z]+\s+Kingdom\./);}' < 419.txt
United Kingdom.
United Kingdom.
United Kingdom. I am writing following an opportunity in my office that
```

[28] "A line of text beginning with a capital letter, followed by 1 or more lower-case letters, followed by one or more whitespace characters, followed by 'Kingdom.'"



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Although it only matches 3 lines, it receives a fitness bonus due to its short length, for a total fitness score of 5.64. Note: to see the spam lines matched by each chromosome, use the following command:

```
perl -e 'while(<>){print if (/<REGEX>/);}' < 419.txt
```

This is a simple Perl script that takes standard input (the file 419.txt), checks each line against a regex, and prints the line of there is a match.

### Generation 2

The 2nd generation contains some chromosomes that have been bred and/or mutated:

| Score | Chromosome |
|---|---|
| 7.82 | `/Kingdom\./` |
| 7.74 | `/the\s+finance/` |
| 7.34 | `/(?:^ive\|^[A-Z][a-z]+\s+Kingdom\.)/` |
| 7.12 | `/(?:^London,England\|^[A-Z][a-z]+\s+Kingdom\.)/` |
| 6.76 | `/(?:^DR\.\s+[A-Z][a-z]+\s+[A-Z][a-z]+\|^[A-Z][a-z]+\s+Kingdom\.)/` |

The chromosome `'/Kingdom\./'` wins this round. It is shorter (and therefore fitter) than `'/A-Z][a-z]+\s+Kingdom\./,'` and also matches an extra line:

```
# perl -e 'while(<>){print if (/Kingdom\./);}' < 419.txt
United Kingdom.
United Kingdom.
United Kingdom. I am writing following an opportunity in my office that
ected by wire transfer via our correspondent bank, Citibank United Kingdom.
```

We also see the first sign of bred chromosomes, showing genes from multiple parents:



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
/(?:^London,England|^[A-Z][a-z]+\s+Kingdom\.)/
```

That matches spam containing ‘London,England’, or ‘United Kingdom.’

## Generation 3

| Score | Chromosome |
|---|---|
| 9.93 | /(?:Kingdom\.\|Johnson\s+Mba\.\|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+)/ |
| 9.57 | /(?:or\s+his\s+.*of\s+kin\.\|^account\s+(?:[a-z]+\s+){2}.*away,\s+an\|the\s+finance)/ |
| 9.47 | /(?:Adjou\.\|Kingdom\.)/ |
| 9.38 | /(?:Diara\.\|the\s+finance)/ |
| 8.88 | /(?:Coulibaly\.\|LOTTERY\s+WINNING\s+\|^deposit)/ |

Bred chromosomes now hold the top 5 scores.

The top-scoring chromosome in generation 3 matches the following spam text:

```
#perl -e 'while(<>){print if(/(?:Kingdom\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+B
ox\s+.*Station,\s+)/);}' < 419.txt
 Barrister Johnson Mba.
PlasmaNet Inc. PO Box 4562, Grand Central Station, New
United Kingdom.
United Kingdom.
United Kingdom. I am writing following an opportunity in my office that
ected by wire transfer via our correspondent bank, Citibank United Kingdom.
#
```

## Generation 4

| Score | Chromosome |
|---|---|
| 25.72 | /(?:LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit)/ |
| 15.85 | /(?:Kingdom\.\|Johnson\s+Mba\.\|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+\|^ive\|^deposit)/ |
| 15.35 | /(?:the\s+finance\|Coulibaly\.\|LOTTERY\s+WINNING\s+\|^deposit)/ |
| 15.35 | /(?:Coulibaly\.\|LOTTERY\s+WINNING\s+\|^deposit\|the\s+finance)/ |
| 13.8 | /(?:MANUEL\|^MISS\s+\|^ive\|LORITA\s+\|Coulibaly\.\|^deposit)/ |



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

The grouped chromosome:

`'/(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/'`

…jumps out to the lead, with a score over 2 ½ times higher than the previous generation's best. This chromosome shows characteristics of many high-scoring chromosomes: multiple short genes that each match large amounts of spam, in (gene1|gene2|gene3…) format.

```
#perl -e 'while(<>){print if (/(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibdeposit
)/);}' < 419.txt
                 YAHOO LOTTERY WINNING NOTIFICATION
INTERNATIONAL LOTTERY PROMO.
International Lottery Program. .
OSA CLAIMS PROCESSING LOTTERY AGENT
SHELL PETROLEUM INTERNATIONAL LOTTERY PROMO NIGERIA
THE YAHOO LOTTERY INTERNATIONAL. INC
WINNING NOTIFICATION LETTER
YAHOO LOTTERY INTL INC
Yahoo Lottery Prize must be claimed no later than 15 days from date of Draw
for this year 2007 Lottery promotion which is organized by
won the Lottery in the 1st category, in four parts. You have been
#
```

## Generation 5

| Score | Chromosome |
|---|---|
| 27.65 | /(?:MANUEL\|^MISS\s+\|^ive\|LORITA\s+\|Coulibaly\.\|^deposit\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit)/ |
| 25.72 | /(?:LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit)/ |
| 19.04 | /(?:Kingdom\.\|Johnson\s+Mba\.\|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+\|^ive\|^deposit\|BANK\s+OF\s+NIGERIA\.\|Coulibaly\.\|LOTTERY\s+WINNING\s+\|MANUEL\|^MISS\s+\|LORITA\s+\|Coulibaly\.)/ |
| 16.17 | /(?:Kingdom\.\|MANUEL\|^MISS\s+\|LORITA\s+\|Coulibaly\.\|BANK\s+OF\s+NIGERIA\.\|Coulibaly\.\|LOTTERY\s+WINNING\s+)/ |
| 15.85 | /(?:Kingdom\.\|Johnson\s+Mba\.\|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+\|^ive\|^deposit)/ |

In generation 5, a new Chromosome takes the lead. In addition to the 'lottery'



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

matches, it adds genes matching the names of alleged widows, such as 'Lorita Manuel,' and 'Mary Coulibaly.'

```
# perl -e 'while(<>){print if (/(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^d
eposit|LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/);}' < 419.txt
                YAHOO LOTTERY WINNING NOTIFICATION
  MY NAME IS MISS LORITA MANUEL,22 YEARS OLD AND THE ONLY
INTERNATIONAL LOTTERY PROMO.
International Lottery Program. .
MISS LORITA MANUEL
MISS LORITA MANUEL.
Mrs.Mary Coulibaly.
OSA CLAIMS PROCESSING LOTTERY AGENT
SHELL PETROLEUM INTERNATIONAL LOTTERY PROMO NIGERIA
THE YAHOO LOTTERY INTERNATIONAL. INC
WINNING NOTIFICATION LETTER
YAHOO LOTTERY INTL INC
Yahoo Lottery Prize must be claimed no later than 15 days from date of Draw
deposit
deposited by my late father.
deposited in CORPORATE SECURITIES CO, a
for this year 2007 Lottery promotion which is organized by
ive
won the Lottery in the 1st category, in four parts. You have been
```

### Generation 6

| Score | Chromosome |
|---|---|
| 29.64 | /(?:^MISS\s+\|^ive\|MANUEL\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|LORITA\s+\|Coulibaly\.\|^deposit)/ |
| 27.82 | /(?:LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit\|or\s+his\s+.*of\s+kin\.\|^account\s+(?:[a-z]+\s+){2}.*away,\s+an\|the\s+finance)/ |
| 27.65 | /(?:MANUEL\|^MISS\s+\|^ive\|LORITA\s+\|Coulibaly\.\|^deposit\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit)/ |
| 26.62 | /(?:or\s+his\s+.*of\s+kin\.\|^account\s+(?:[a-z]+\s+){2}.*away,\s+an\|the\s+finance\|MANUEL\|^MISS\s+\|^ive\|LORITA\s+\|Coulibaly\.\|^deposit\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit)/ |
| 25.72 | /(?:LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit)/ |

Generation 5's winner survives with the lead in generation 6. Three new chromosomes take slots 2-4.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

### Generation 7

| Score | Chromosome |
|---|---|
| 29.64 | /(?:^MISS\s+\|^ive\|MANUEL\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|LORITA\s+\|Coulibaly\.\|^deposit)/ |
| 29.44 | /(?:^MISS\s+\|^ive\|MANUEL\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|LORITA\s+\|Coulibaly\.\|^deposit\|the\s+finance\|Coulibaly\.\|LOTTERY\s+WINNING\s+\|^deposit)/ |
| 27.82 | /(?:LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit\|or\s+his\s+.*of\s+kin\.\|^account\s+(?:[a-z]+\s+){2}.*away,\s+an\|the\s+finance)/ |
| 27.65 | /(?:MANUEL\|^MISS\s+\|^ive\|LORITA\s+\|Coulibaly\.\|^deposit\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit)/ |
| 27.27 | /(?:the\s+finance\|Coulibaly\.\|LOTTERY\s+WINNING\s+\|^deposit\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit)/ |

Once again, generation 5's winner holds on to the lead in generation 7. A new chromosome takes the #2 slot.

### Generation 8

| Score | Chromosome |
|---|---|
| 30.24 | /YOUR\s+[A-Z]+\s+[A-Z]+/ |
| 29.64 | /(?:^MISS\s+\|^ive\|MANUEL\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|LORITA\s+\|Coulibaly\.\|^deposit)/ |
| 29.44 | /(?:^MISS\s+\|^ive\|MANUEL\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|LORITA\s+\|Coulibaly\.\|^deposit\|the\s+finance\|Coulibaly\.\|LOTTERY\s+WINNING\s+\|^deposit)/ |
| 27.82 | /(?:LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit\|or\s+his\s+.*of\s+kin\.\|^account\s+(?:[a-z]+\s+){2}.*away,\s+an\|the\s+finance)/ |
| 27.65 | /(?:MANUEL\|^MISS\s+\|^ive\|LORITA\s+\|Coulibaly\.\|^deposit\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|Coulibaly\.\|^deposit)/ |

A new chromosome takes the lead, an automatic chromosome matching capital 'YOUR', followed by two capitalized words (each word followed by at least one whitespace character.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

This chromosome receives a high fitness score because it is short, and it matches emails written in ALL CAPS (as many 419 scams are, to imply urgency), asking 'YOU' to do something.

The following spam lines are matched:

```
# perl -e 'while(<>){print if (/YOUR\s+[A-Z]+\s+[A-Z]+/);}' < 419.txt
  I AM EXPECTING YOUR IMMEDIATE REPLY.
  I AM HAPPY TO REQUEST FOR YOUR ASSISTANCE BECAUSE I BELIVE THAT YOU ARE NOT GO
ING TO BETRAY THE TRUST WHICH I AM GOING TO LAY ON YOU.
  I WANT YOU TO ASSIST ME TO TRANSFER THIS MONEY TO YOUR COUNTRY FOR INVESTMENT
PURPOSES,SO I CAN FUTHER MY EDUCATION AND LIVE A NEW LIFE.
1. YOUR FULL NAME
2.YOUR HOME ADDRESS.
3.YOUR CURRENT HOME TELEPHONE NUMBER.
4.YOUR CURRENT OFFICE TELEPHONE.
AND AT THE SAME  TIME  ENSURING THAT YOUR BANK ACCOUNT IS STILL  OPEN TO
BASED ON THAT YOU HAVE BEEN ADVISED NOT TO RESPOND TO ANY MAIL OR FAX CONCERNING
 YOUR FUND FROM ANY BODY WHO CLAIM TO BE OR YOUR AGENT AS YOUR PAYMENT  IS NOW R
EADY TO BE  TRANSFERED IMMEDAITELY  MY OFFICE  CONFIRM RECEIPT OF THE FOLLOWINGS
.
HIS DIRECT EMAIL ADDRESS BELLOW AND RECONFIRM TO HIM  YOUR CONTRACT NUMBER
I AWAIT YOUR URGENT REPLY.
OF CLAIM TO MY OFFICE ON YOUR BEHALF FOR THE RELEASE OF YOUR PAYMENT OF TEN
RECEIVE  YOUR APPROVED  CONTRACT  FUNDS
RECEIVING BANKING INFORMATION AND YOUR RESIDENCIAL ADDRESS AND YOUR DIRECT
WHICH IS YOUR SUBJECT PAYMENT CODE AND APPROVED CONTRACT AMOUNT AND YOUR
YOUR ADVICE TO CONTACT THE  DIRECTOR FORIEGN OPERATION,DR.IGNATIUS IMALLA ON
```

## Generation 9

| Score | Chromosome |
|---|---|
| 39.75 | /(?:Coulibaly\.\|LOTTERY\s+WINNING\s+\|^deposit\|the\s+finance\|YOUR\s+[A-Z]+\s+[A-Z]+)/ |
| 31.17 | /(?:YOUR\s+[A-Z]+\s+[A-Z]+\|^ive\|^[A-Z][a-z]+\s+Kingdom\.\|Coulibaly\.\|Johnson\s+Mba\.\|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+\|^deposit\|MANUEL\|^MISS\s+\|^ive\|LORITA\s+\|Coulibaly\.\|^deposit)/ |
| 30.24 | /YOUR\s+[A-Z]+\s+[A-Z]+/ |
| 29.64 | /(?:^MISS\s+\|^ive\|MANUEL\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|LORITA\s+\|Coulibaly\.\|^deposit)/ |



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
       /(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|
       LORITA\s+|Coulibaly\.|^deposit|the\s+finance|Coulibaly\.|
29.44  LOTTERY\s+WINNING\s+|^deposit)/
```

Last generation's winner is bred with another strong chromosome, chosen via Roulette Wheel selection, resulting in a chromosome that matches the 'YOUR..' capitalized text, 'Mrs' Coulibaly, the phrase 'LOTTERY WINNING', lines beginning with 'deposit' and the phrase 'the finance'.

Here are the lines of spam generation 9's winning chromosome matches:

© SANS Institute 2007, Author retains full rights.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
# perl -e 'while(<>){print if (/(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the
\s+finance|YOUR\s+[A-Z]+\s+[A-Z]+)/);}' < 419.txt
                 YAHOO LOTTERY WINNING NOTIFICATION
  I AM EXPECTING YOUR IMMEDIATE REPLY.
  I AM HAPPY TO REQUEST FOR YOUR ASSISTANCE BECAUSE I BELIVE THAT YOU ARE NOT GO
ING TO BETRAY THE TRUST WHICH I AM GOING TO LAY ON YOU.
  I WANT YOU TO ASSIST ME TO TRANSFER THIS MONEY TO YOUR COUNTRY FOR INVESTMENT
PURPOSES,SO I CAN FUTHER MY EDUCATION AND LIVE A NEW LIFE.
1. YOUR FULL NAME
2.YOUR HOME ADDRESS.
3.YOUR CURRENT HOME TELEPHONE NUMBER.
4.YOUR CURRENT OFFICE TELEPHONE.
AND AT THE SAME  TIME  ENSURING THAT YOUR BANK ACCOUNT IS STILL  OPEN TO
BASED ON THAT YOU HAVE BEEN ADVISED NOT TO RESPOND TO ANY MAIL OR FAX CONCERNING
 YOUR FUND FROM ANY BODY WHO CLAIM TO BE OR YOUR AGENT AS YOUR PAYMENT  IS NOW R
EADY TO BE  TRANSFERED IMMEDAITELY  MY OFFICE  CONFIRM RECEIPT OF THE FOLLOWINGS
.
HIS DIRECT EMAIL ADDRESS BELLOW AND RECONFIRM TO HIM  YOUR CONTRACT NUMBER
I AWAIT YOUR URGENT REPLY.
I have authorized the finance house where I deposited my money to issue you
Mrs.Mary Coulibaly.
OF CLAIM TO MY OFFICE ON YOUR BEHALF FOR THE RELEASE OF YOUR PAYMENT OF TEN
RECEIVE  YOUR APPROVED  CONTRACT  FUNDS
RECEIVING BANKING INFORMATION AND YOUR RESIDENCIAL ADDRESS AND YOUR DIRECT
WHICH IS YOUR SUBJECT PAYMENT CODE AND APPROVED CONTRACT AMOUNT AND YOUR
YOUR ADVICE TO CONTACT THE  DIRECTOR FORIEGN OPERATION,DR.IGNATIUS IMALLA ON
deposit
deposited by my late father.
deposited in CORPORATE SECURITIES CO, a
forwarded instruction to the finance house on your behalf to send the bank
inform the finance
the finance company contacted me, as his attorney to provide his next of
#
```

## Generation 10

| Score | Chromosome |
|---|---|
| 50.58 | /(?:YOUR\s+[A-Z]+\s+[A-Z]+\|^MISS\s+\|^ive\|MANUEL\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|LORITA\s+\|Coulibaly\.\|^deposit)/ |
| 39.75 | /(?:Coulibaly\.\|LOTTERY\s+WINNING\s+\|^deposit\|the\s+finance\|YOUR\s+[A-Z]+\s+[A-Z]+)/ |
| 31.17 | /(?:YOUR\s+[A-Z]+\s+[A-Z]+\|^ive\|^[A-Z][a-z]+\s+Kingdom\.\|Coulibaly\.\|Johnson\s+Mba\.\|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+\|^deposit\|MANUEL\|^MISS\s+\|^ive\|LORITA\s+\|Coulibaly\.\|^deposit)/ |
| 30.24 | /YOUR\s+[A-Z]+\s+[A-Z]+/ |
| 29.64 | /(?:^MISS\s+\|^ive\|MANUEL\|LOTTERY\s+\|Lottery\s+\|WINNING\s+\|LORITA\s+\|Coulibaly\.\|^deposit)/ |



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

In the final generation, a new chromosome takes the lead, scoring over 50. It combines the genes from generation 9's winner, adding the gene 'MANUEL' (matching lines containing 'MISS LORITA MANUEL', and the gene 'Lottery\s+' (matching any line containing 'Lottery,' followed by whitespace.

Here are the results from the winner of the final generation:

© SANS Institute 2007, Author retains full rights.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
# perl -e 'while(<>){print if (/(?:YOUR\s+[A-Z]+\s+[A-Z]+|^MISS\s+|^ive|MANUEL|L
OTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^deposit)/);}' < 419.txt
                  YAHOO LOTTERY WINNING NOTIFICATION
  I AM EXPECTING YOUR IMMEDIATE REPLY.
  I AM HAPPY TO REQUEST FOR YOUR ASSISTANCE BECAUSE I BELIVE THAT YOU ARE NOT GO
ING TO BETRAY THE TRUST WHICH I AM GOING TO LAY ON YOU.
  I WANT YOU TO ASSIST ME TO TRANSFER THIS MONEY TO YOUR COUNTRY FOR INVESTMENT
PURPOSES,SO I CAN FUTHER MY EDUCATION AND LIVE A NEW LIFE.
  MY NAME IS MISS LORITA MANUEL,22 YEARS OLD AND THE ONLY
1. YOUR FULL NAME
2.YOUR HOME ADDRESS.
3.YOUR CURRENT HOME TELEPHONE NUMBER.
4.YOUR CURRENT OFFICE TELEPHONE.
AND AT THE SAME  TIME  ENSURING THAT YOUR BANK ACCOUNT IS STILL  OPEN TO
BASED ON THAT YOU HAVE BEEN ADVISED NOT TO RESPOND TO ANY MAIL OR FAX CONCERNING
 YOUR FUND FROM ANY BODY WHO CLAIM TO BE OR YOUR AGENT AS YOUR PAYMENT  IS NOW R
EADY TO BE  TRANSFERED IMMEDAITELY  MY OFFICE  CONFIRM RECEIPT OF THE FOLLOWINGS
.
HIS DIRECT EMAIL ADDRESS BELLOW AND RECONFIRM TO HIM  YOUR CONTRACT NUMBER
I AWAIT YOUR URGENT REPLY.
INTERNATIONAL LOTTERY PROMO.
International Lottery Program. .
MISS LORITA MANUEL
MISS LORITA MANUEL.
Mrs.Mary Coulibaly.
OF CLAIM TO MY OFFICE ON YOUR BEHALF FOR THE RELEASE OF YOUR PAYMENT OF TEN
OSA CLAIMS PROCESSING LOTTERY AGENT
RECEIVE  YOUR APPROVED  CONTRACT  FUNDS
RECEIVING BANKING INFORMATION AND YOUR RESIDENCIAL ADDRESS AND YOUR DIRECT
SHELL PETROLEUM INTERNATIONAL LOTTERY PROMO NIGERIA
THE YAHOO LOTTERY INTERNATIONAL. INC
WHICH IS YOUR SUBJECT PAYMENT CODE AND APPROVED CONTRACT AMOUNT AND YOUR
WINNING NOTIFICATION LETTER
YAHOO LOTTERY INTL INC
YOUR ADVICE TO CONTACT THE  DIRECTOR FORIEGN OPERATION,DR.IGNATIUS IMALLA ON
Yahoo Lottery Prize must be claimed no later than 15 days from date of Draw
deposit
deposited by my late father.
deposited in CORPORATE SECURITIES CO, a
for this year 2007 Lottery promotion which is organized by
ive
won the Lottery in the 1st category, in four parts. You have been
```

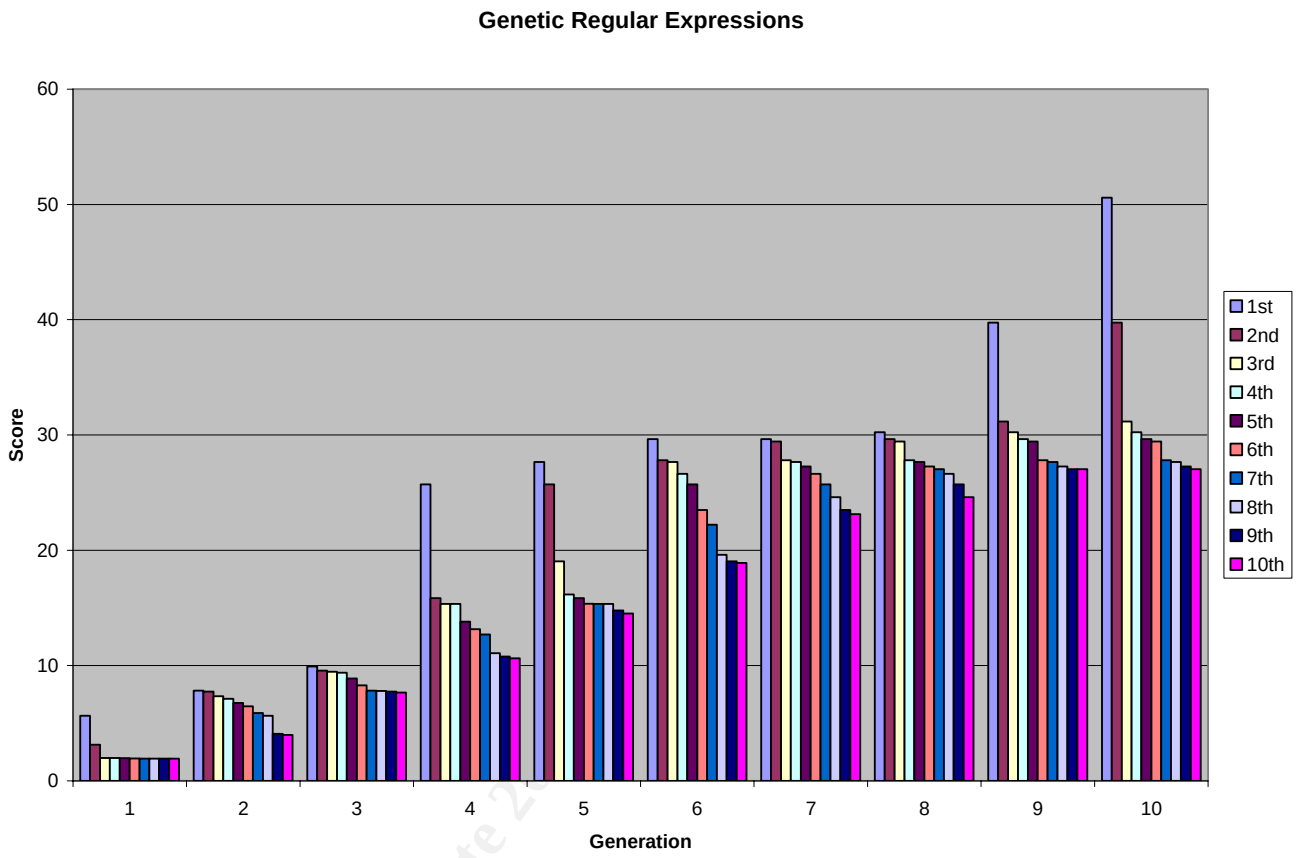
## Summary

Here is a summary graph, showing the performance of the top-10 scoring chromosomes, over 10



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

generations




## Sample SpamAssassin Rule

This SpamAssassin rule is based on the winning genregex.pl rule:

```
body GENETIC_REGEX_1            /(?:YOUR\s+[A-Z]+\s+[A-
Z]+|^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA
\s+|Coulibaly\.|^deposit)/

score GENETIC_REGEX_1           1.0

describe GENETIC_REGEX_1        'Advanced Fee' spam rule written
by genregex.pl
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

## Efficiency of genetic regular expressions

### Size of the corpora

The proof-of-concept example run described above was run on a FreeBSD 6.2 server with a 3.2 gigahertz Xeon processor, and 2 gigs of memory.

The spam corpus contained 1155 unique lines, and the ham corpus contained 14799.

The run described in this paper generated the following statistics:

```
Comparisons: 8088.  Comparisons/sec:38.8846153846154. Time
taken was 208 seconds
```

208 seconds is a reasonable amount of time for a proof-of-concept script, but much larger corpora are required for truly useful operational results. The ham corpora did not contain the word 'Lottery,' for example, but it's a word that does exist in non-spam email.

Much larger corpora (10 times+) would be useful for an operational environment (avoiding words like 'Lottery'), which would result in program execution time of hours or more.

## A note on 'ors'

Large amounts of 'ors' ('|') become inefficient, as the regex engine must rescan a given multiple times for each 'or'.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

This is the winning chromosome:

```
/(?:YOUR\s+[A-Z]+\s+[A-Z]+|^MISS\s+|^ive|MANUEL|
LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|
Coulibaly\.|^deposit)/
```

The regex engine must take a line of text, and run the following (simplified) checks for the winning chromosome:

```
1. Scan the entire line for YOUR\s+[A-Z]+\s+[A-Z]+
2. Scan the beginning of the line (only) for MISS\s+
3. Scan the beginning of the line (only) for ive
4. Scan the entire line for MANUEL
5. Scan the entire line for LOTTERY\s+
6. Scan the entire line for Lottery\s+
7. Scan the entire line for WINNING\s+
8. Scan the entire line for LORITA\s+|
9. Scan the entire line for Coulibaly\.
10.    Scan the beginning of the line (only) for deposit
```

'Anchored' searches (using '^', for example) are faster than unanchored searches, which must search the entire line for matching text. In the example run, the fittest chromosomes contained lots of 'or' operations, resulting in slower matches.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

## 6. Conclusion

Genetic regular expressions effectively match spam, and show improvement generation-to-generation. The winning chromosome from the 10th generation was over 9 times as effective as the winning chromosome from the first.

Genetic Regular expressions exhibit the following characteristics, as described by Hiu Wong: 'The basic techniques of the GAs are designed to simulate processes in natural systems necessary for evolution, specially those follow the principles first laid down by Charles Darwin of "survival of the fittest."'[29]

Genetic regular expressions leverage the Genetic Algorithm concepts of fitness, crossover, and mutation to evolve chromosomes across generations to find a superior solution.

---

[29] Wong, Hiu. *Introduction to Genetic Algorithms. Surveys and Presentations in Information Systems Engineering (SURPRISE) Journal 96 Volume 4.* URL: http://www-dse.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

# 7. Further study

### Evolving Regex Primitives

A promising field of study is evolving regex primitives used by genetic regular expressions. Certain regex design decisions were made in this paper, such as using actual lines of spam as the basis for chromosomes, joining multiple genes with 'or' operations, ensuring grouping parentheses always match, etc. These design decisions were made due to their practical and efficient application towards the problem domain.

A 'primitive approach' would break regexes down further, creating a 'primordial soup' of characters, including all literal characters that may be found in spam, and all regex metacharacters. Regexes could be built from this 'soup.' Initial generations would mostly comprise of nonsense strings of illegal regexes (unbalanced parentheses, anchor metacharacters in the middle of a string, etc) which would quickly die.

The approach would require building a regex analyzer that could not only determine 'legal/illegal' status of regex syntax, but also assign a syntax score: gibberish would receive a low score, and a regex that is one parenthesis away from 'legal' would receive a higher score.

Illegal regexes would be allowed to survive in early generations (until sufficient legal



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

regexes out-survive them), to help get over the initial 'legal regex' survival threshold ('crawl before you walk)'. Ham-matching regexes would also survive (for a limited number of generations), until they are 'beaten' by spam-matching regexes.

Here is the logic for such an approach

```
1. Create regex primordial soup
2. Loop until done:
     a. Generate regexes from the 'soup'
     b. Assign a regex syntax score (0 to 1, graded from
        gibberish to legal syntax)
     c. Check for ham matches (score 1 if match)
     d. Check and score for spam matches (score 2+ if match)
     e. Keep the top-scoring 3rd
     f. Create children based on any survivors
     g. Mutate children (deleting genes, or adding random genes
        from the soup)
```

It would probably take hundreds or more generations (representing days of 3 ghz Xeon-equivalent CPU time) to evolve useful regexes, but, by leveraging Darwinian survival mechanisms, the regexes themselves could prove more powerful that those generated in POC code in this paper.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

# 8. Appendices

Proof-of-concept code, including the spam and ham corpora used in this paper, may be downloaded from: http://files.ericconrad.com/genregex.tgz

The author may be reached at eric@ericconrad.com

## Appendix A 'autoregex.pl,' POC Perl Script demonstrating Automatic Regular Expressions

```
#!/usr/bin/perl

my $regex="";

open(HAMTXT,"<hams.txt")||die;
@hamlines = <HAMTXT>;
close(HAMTXT);

open(SPAMTXT,"<spams.txt")||die;
@spamlines = <SPAMTXT>;
close(SPAMTXT);

foreach(@spamlines){
  chomp;
  my $newstring="";
  #if (/^\t/){
  #  $newstring="^\\t";
  #}
  $len=length($_);
  $count=0;
  $wordcount=0;
  @word="";
  @line=split(//);
  while($count<$len){
    # Break string into a series of tokens
    # each word is a token, and each symbol (like "@" or "<") is a token
    # ":" is a token, unless it's in the format /^\w:\s/ (Matches headers like
/^Subject: /)
    if(($line[$count]=~/[A-Za-z0-
9]/)||(($line[$count]=~/:/)&&($line[$count+1]=~/\s/))){
      $wordcount++ if (($count!=0)&&($line[$count-1]!~/[A-Za-z0-9]/));
      $word[$wordcount].=$line[$count];
    }
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
    else{
      $wordcount++ if ($count!=0);
      # Escape all metacharacters we want to treat as literals
      if ($line[$count]=~/[{}\[\]()^$\.|*+?\\]/){
        $word[$wordcount]="\\";
      }
      $word[$wordcount].=$line[$count];
    }
    $count++;
  }
   $first=1;
   foreach(@word){
      $wordlen=length($_);
      if (($first)&&(/^[-\w]+:$/)){  # Match strings like  /^Subject: /
        $newstring="^$_";
        $first=0;
      }
      elsif ($wordlen>1){        # If token is a word (>1 char)
        #if (/^[0-9]+$/){         # Numbers
        #  $newstring.="[0-9]+";
        if (/^\d+$/){         # Numbers
          $newstring.="\\d+";
        }
        elsif (/^[A-F0-9]+$/){   # Hex with caps
          $newstring.="[A-F0-9]{$wordlen}";
          #$newstring.="[A-F0-9]+";
        }
        elsif (/^[a-f0-9]+$/){   # Hex with lowercase
          $newstring.="[a-f0-9]{$wordlen}";
          #$newstring.="[a-f0-9]+";
        }
        elsif (/^(com|net|org|edu|biz|info|us)$/){      # lowercase letters
          $newstring.="(com|net|org|edu|biz|info|us)";
        }
        elsif (/^[a-z]+$/){      # lowercase letters
          $newstring.="[a-z]+";
        }
        elsif (/^[A-Z]+$/){      # Capital letters
          $newstring.="[A-Z]+";
        }
        elsif (/^(Mon|Tue|Wed|Thu|Fri|Sat|Sun)$/){ # Days of the week
          $newstring.="(Mon|Tue|Wed|Thu|Fri|Sat|Sun)";
        }
        elsif (/^(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)$/){  # Months of
the year
          $newstring.="(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)";
        }
        elsif (/^[A-Z][a-z]+$/){      # Words With The First Letter Capitalized
          $newstring.="[A-Z][a-z]+";
        }
        else{
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
          $newstring.="$_";
        }
      }
      else{  # Token is 1 character
        if (/[0-9]/){
          $newstring.="[0-9]";
        }
        elsif (/[A-Z]/){
          $newstring.="[A-Z]";
        }
        elsif (/[a-z]/){
          $newstring.="[a-z]";
        }
        elsif (/\t/){
          $newstring.="\\t";
        }
        else {
          $newstring.="$_";
        }
      }
    }
  if ($newstring =~ /  +/){
    #print "$newstring\n";
    $newstring =~ s/  +/  \+/g;
    #print "$newstring\n";
  }
  $candidates{$newstring}=1;
  }

# Delete any regext that match ham
while ( ($regex, $count) = each %candidates ) {
  foreach(@hamlines){
    if ((/$regex/g)&&(!($hamhit{$regex}))){
      $hamhit{$regex}=1;
      #print "MISS: $regex\n";
    }
  }
}

# Rank remaining regex by # of spams matched
while ( ($regex, $count) = each %candidates ) {
  foreach(@spamlines){
    if ((/$regex/g)&&(!($hamhit{$regex}))){
      $regexes{$regex}++;
      #print "HIT: $regex\n";
    }
  }
}

@sorted = sort { $regexes{$a} <=> $regexes{$b} } keys %regexes;
foreach (@sorted){
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
        print "$regexes{$_} /$_/\n";
}
```

## Appendix B: 'genregex.pl', Proof-of-concept Perl Script demonstrating Genetic Regular Expressions

```
#!/usr/bin/perl -w

# genregex.pl  v. 0.1
# Eric Conrad, September 2007
# genregex@ericconrad.com

# This program is free software, licensed under the GNU GPL, >=2.0.
# This software comes with absolutely NO WARRANTY. Use at your own risk!

use strict;

my %regexes;         # Hash that tracks all regexes
my %scores;          # Hash that tracks scores of chromosomes
my @fittest;         # Array containing top third chromosomes, sorted by score

my $start = time();  # Start of prgram execution
my @hamlines;        # array containing ham text
my @spamlines;       # array containing spam text
my $generations=10;  # Number of generations to evolve
my $generation=0;    # The current generation
my %hamhit;          # Hash that tracks whether a regex matched ham
my @survivors = ();  # The survivors chromosomes AoA
my %survivorhash;    # hash of references to the survivors AoA, key is the regex
itself
my $score=0;         # Weighted score of a regex (matches * multiplier)
my $comparisons=0;   # Used to track # of regex comparisons

# Program logic:
#
# Read the spam and ham corpora
# Divide spam corpus into 10 slices
# Loop for 10 generations:
#   Create new chromosomes based on the next slice of spam
#   Create regex strings based on the chromosomes
#   Rank the chromosomes
#   Keep 1/3rd based on the highest score
#   Create a child for each of the current third

readcorpus(\@hamlines,\@spamlines);    # Read the spam and ham corpora

fisher_yates_shuffle(\@spamlines);     # Shuffle the spam lines randomly
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
my $spamlines=scalar(@spamlines);         # Number of lines of spam
my $fract=int($spamlines/$generations);   # Percentage of spam to process each
generation
my $thirtieth=int($spamlines/30);         # 10 generations, 1/3 of each is a
thirtieth

# Main program loop
while($generation <$generations ){
  print "Generation $generation #######################\n";
  my $offset=int($generation*$fract);     # Offset to gauge which chunk of spam is
being processed
  create_chromosomes($offset,($offset+$fract),\@survivors);  # Create new
chromosomes based off the spam
  %survivorhash=stringify(\@survivors); # Create regex strings based on @survivors
  misses(\@hamlines);                   # Ignore any survivors that match ham
  %regexes=hits(\@spamlines);           # tabulate hits per survivors
  %scores=scoreregexes(%regexes);       # Reward shorter survivors (and punish
longer ones)
  @fittest=fittest(%scores);            # Grab the top third scores
  @survivors=breed(\@fittest);          # Breed chromosomes based on the top third
  $generation++;
}

# end timer
my $end = time();

# report
print "Comparisions: $comparisons.  Comparisons/sec:", ($comparisons/($end -
$start)), ". Time taken was ", ($end - $start), " seconds";

sub breed{
  my $fref=shift;
  my @fittest=@{$fref};
  my @survivors;
  my @child;
  my $counter=0;            #temporary counter
  my @breedingpool=();
  my $breedref;
  my $cutoff=10;
  my $fitcounter=0;

  # Populate the breeding pool.  Higher scoring regexes get more chances to breed
  # Use 'Roulette wheel' selection, based on the regex score.  A score of 1 gives
  # 1 slot in the breeding pool, score of 3 gives 3 slots, etc.

  # A slot in the pool is a pointer to the survivorhash.  A regex with a score of
10
  # gets 10 pointers.

  foreach (@fittest){
    $counter=$fitcounter;
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
    $fitcounter+=int($scores{$_});
    while($counter<$fitcounter){
      $breedingpool[$counter]=\@{$survivorhash{$_}};
      $counter++;
    }
  }
  $counter=0;
  foreach (@fittest){
    printf("%2.2f /%s/\n",$scores{$_},$_) if ($counter < $cutoff);
    $counter++;
    my $index = int (rand @breedingpool);   # Select a chromosome from the breeding
pool
    $breedref = $breedingpool[$index];
    @child=crossover(\@{$survivorhash{$_}},$breedref);     # Create a child
chromosome based on the current survivor
    optimize(\@child,0);                      # Optimize the chromosome.
    push @survivors, [ @child ];              # Save the child chromosome
    push @survivors, [ @{$survivorhash{$_}} ]; # Also save the current survivor
  }
  return(@survivors);
}

sub fittest{
  my @sorted;
  my $sortlines;
  %scores=(@_);

  @sorted = sort { $scores{$b} <=> $scores{$a} } keys %scores;
  splice(@sorted,$thirtieth);      # Keep the best-scoring third
  return(@sorted);
}

sub scoreregexes{
  my %regexes=@_;
  my $length=0;
  my $multiplier=0;
  my $regex;
  my $matches;

  while(($regex, $matches) = each(%regexes)) {
    $length=length($regex);
    #$length=length($regex)/2;
    $multiplier=(1+((200-$length)/200));
    #$multiplier=1 if ($length>=100);
    $score=($multiplier * $matches);
    $scores{$regex}=$score;
  }
  return(%scores);
}

sub stringify{
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
  # Generate a tring regex based on a chromosome
  my $sref=shift;
  my @survivors=@{$sref};
  my $cref;          # Reference to a chromosome
  my $regex="";
  my %survivorhash=();

  for $cref ( @survivors ) {
    $regex="";
    foreach(@$cref){
      $regex.="$_";                      # Create a regex string based on the
chromosome
    }
    $survivorhash{$regex}=$cref;  # Build hash of references to the survivors array
of arrays, key is the regex itself
  }
  return(%survivorhash);
}

sub crossover{
  my $motherref =shift (@_);
  my $fatherref =shift (@_);

  my @mother=@{$motherref};
  my @father=@{$fatherref};

  my $mlength=scalar(@mother);    # Length of mother
  my $flength=scalar(@father);    # Length of father

  my @child;
  my $clength;

  if (int(rand(2))==1){      # 50% chance 2 chromosomes are broken down to their
genes, and combined in (a | y | b | z ) format
    push @mother,@father;                     # Going to swap between mother and
child for some array manipulations
    @child = grep(!/\)|\(|\||\.\*/, @mother);   # Remove genes with (,),|, { or .*.
    # Grab unique genes and sort them.  Sort avoids (a | b) and (b | a) appearing
to be 2 different genes
    my %hash   = map { $_, 1 } @child;
    @child = sort { $hash{$a} cmp $hash{$b} } keys %hash;
    $clength=scalar(@child);
    if ($clength>1){
      #mutate(\@child) if (int(rand(2))==1);
      mutate(\@child);
      $clength=scalar(@child)-1;  # Length may have changed
      while($clength>0){
        splice (@child,$clength,0,"|");
        $clength--;
      }
      unshift @child,"(?:";
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
      push @child,")";
    }
  }
  else{

    # 4 cases of mutation:
    #   1. mother + father
    #   2. (?:mother)+ father
    #   3. mother + (?:father)
    #   4. (?:mother) + (?:father)

    # There is probably simpler/cleaner logic to perform the following.  The goal
is to avoid costly nested groups,
    # and keep things in "(foo|bar|baz)" format, and not "((foo|bar)|baz)" format

    # In mutation occurs via deleting a random amount of genes.  Depending on the
grouping characteristsics, a father, mother, or both
    # may have genes deleted before they are bred.  If oth parents are pre-grouped,
no deletion occurs

    # TODO: write a more robust deletion/mutation subroutine that handles all
cases, and mutates all types of children
    # as opposed to pre-mutation some parents only

    if (($father[$flength-1] eq ")")&& ($mother[$mlength-1] eq ")")){  # both
mother and father are groups
      pop @mother;            # Remove the trailing ")"
      push @mother,"|";
      shift @father;       # Remove the leading "(?:"
      push @mother,@father;
      removeduplicates(\@mother);
    }
    elsif ($mother[$mlength-1] eq ")"){ # mother is already grouped, father is not
      deletegenes(\@father);
      pop @mother;            # Remove the trailing ")"
      push @mother,"|";
      push @mother,@father;
      push @mother,")";
      removeduplicates(\@mother);
    }
    elsif ($father[$flength-1] eq ")"){ # father is grouped, mother is not
      deletegenes(\@mother);
      unshift @mother,"(?:";
      push @mother,"|";
      shift @father;       # Remove the leading "(?:"
      push @mother,@father;
      removeduplicates(\@mother);
    }
    else{                     # Neither is grouped
      deletegenes(\@mother);
      deletegenes(\@father);
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
      unshift @mother,"(?:";
      push @mother,"|";
      push @mother,@father;
      push @mother,")";
    }
    @child=@mother;
  }
  return(@child);
}

sub deletegenes{
  # TODO: Combine with mutate(), below
  my $arrayref=shift;
  my @chromosome=@{$arrayref};
  my $length=scalar(@chromosome);
  my $random=int(rand($length)); # Random integer, keyed off parent1 length
  my $amount=int(rand($length-$random));
  splice(@chromosome,$random,$amount);  # delete a random number of genes
  splice(@chromosome,$random,0,".*") if (($random>0)&&($random<$length));
}

sub mutate{
  # TODO: Combine with deletegrenes(), above
  my $childref=shift;
  my @chromosome=@$childref;
  my $mutations=4; # Number of different mutations available
  my $amount=0;                  # Amount of genes to delete
  my $choice;
  my $length2=scalar(@chromosome);
  my $random=int(rand($length2)); # Random integer, keyed off parent1 length

  fisher_yates_shuffle(\@chromosome);
  $amount=int(rand($length2-$random));
  splice(@chromosome,$random,$amount);  # delete a random number of genes

}

sub fisher_yates_shuffle {
  # Code from the Perl FAQ 'How do I suffle an array randomly?'
  # http://perldoc.perl.org/perlfaq4.html#How-do-I-shuffle-an-array-randomly%3f
  my $deck = shift;  # $deck is a reference to an array
  my $i = @$deck;
  while (--$i) {
      my $j = int rand ($i+1);
      @$deck[$i,$j] = @$deck[$j,$i];
  }
}

sub genregex{
  my $linesref=shift(@_);
  my @currentlines=@{$linesref};
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
  my @chromosomes; # Array of chromosome arrays
  my @chromosome;  # One chromosome
  my @spamline;    # Line of spam text

  foreach(@currentlines){
    #print "GEN: $_\n";
    chomp;
    s/\s+/ /g;                    # Reduce repeated whitespace to 1 space
    @spamline=split(" ");         # Split words into an array
    @chromosome=create_genes(@spamline);  # Create chromosomes based on each word
    optimize(\@chromosome,1);         # Optimize the chromosome.  Pass by
reference; chromosome will be altered
    push @chromosomes, [ @chromosome ];
  }
  return(@chromosomes);
}

sub optimize {
  # Optimize the chromosomes by grouping repeated genes, and deleting redundant
genes (like .*.*)
  # Altering the passed array, so all array operations must be made by reference

  my $repeats=0;    # Number of repeated genes (to be grouped)
  my $baseregex=""; # Regex before grouping

  my $cref = shift(@_);
  my $first = shift(@_);

  my $wordcount= (scalar @{$cref})-1; # Number of words in the array.  array begins
at offset 0, not 1
  my $current=$wordcount;  # Current position of genes array

  # Take 2 passes; one to add whitespace and delete repeated ".*"; the 2nd to
combine identical
  # repeated genes.  Could probably be done more efficiently in one pass.

  # 1st pass
  while($current>=0){
    if (($first)&&($current!=$wordcount)){ # Append whitespace unless it's the last
genes or ".*"
      $cref->[$current].="\\s+" unless ($cref->[$current] eq ".*");
    }
    # Remove repeated ".*"
    while (($current)&&($cref->[$current] eq ".*")&&($cref->[$current-1]) eq ".*"){
      splice(@{$cref},$current,1);
      $current--;
    }
    $current--;
  }
  $wordcount= (scalar @{$cref})-1; # may have changed due to splice
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
  $current=$wordcount;
  # 2nd pass
  while($current>0){
    $baseregex=$cref->[$current];
    $repeats=0;
    # If there are repeated genes, combine them in (genes){count} format
    while (($current)&&($cref->[$current] eq $cref->[$current-1])){
      $repeats++;
      splice(@{$cref},$current,1);
      $current--;
    }
    if ($repeats){
      $repeats++;
      $cref->[$current]="(?:$baseregex){$repeats}";      # ?: is ugly, but avoids
capturing, which is slow
    }
    $current--;
  }
  if ($cref->[0] eq ".*"){  # Remove leading ".*"
    splice(@{$cref},0,1)
  }
  else{
    # Append a "^" on the first run, and skip regexes that began with ".*"
    $cref->[0]="^$cref->[0]" if ($first);
  }

  $wordcount= (scalar @{$cref})-1; # may have changed due to splice
  # Remove trailing ".*" (redundant).  Check now, after multiple contiguous ".*"s
have been consolidated
  if ($cref->[$wordcount]){
    splice(@{$cref},$wordcount,1) if (($cref->[$wordcount]) eq ".*");
  }
}

sub removeduplicates{

  # combines duplicate genes, changing:
  #     /(?:FOO|You ([a-z]+\s+){2} eaten by a grue|FOO|BAR)/
  # to: /(?:FOO|You ([a-z]+\s+){2} eaten by a grue|BAR)/
  # Simple shortcuts like creating a unique hash don't help, as 'eaten by a grue'
is 4 genes that must remain in order
  # Logic is 2 loops: start with gene1, and search for dups in gene1+X
  # Then:                     gene2, and search for dups in gene2+X
  # Etc.
  #
  # If a dup is found, delete the 2nd matching gene, plus the previous "|".
  # There is probably a cleaner way to do this
  #
  # TODO: Some dups are still slipping through, probably due to a bug in the below
code.
  #
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
  my $cref = shift(@_);
  my @chromosome=@{$cref};
  my $count=1;
  my $length=scalar(@chromosome);
  my @pipes=grep("\|",@chromosome);
  if ((@pipes)&&($chromosome[0] eq "(?:")){
    while($count<$length){
      my $first=$chromosome[$count];
      $count++;
      my $count2=$count;
      while($count2<$length){
        $count2++;
        if (($count2<$length)&&($chromosome[$count2] eq "|")){
          $count2++;
          if (($count2<$length)&&($first eq "$chromosome[$count2]")){
            my $offset=$count2-1;
            $count2++;
            if (($count2<$length)&&(($chromosome[$count2] eq
"|")||($chromosome[$count2] eq ")")) ){
              splice(@chromosome,$offset,2);
              $length=scalar(@chromosome);    # Reset length; chromosome is shorter
            }
          }
        }
      }
    }
  }
}

sub create_genes {
  #
  # TODO: code creates some infrequent 'strict' warnings, need to be fixed
  #
  my @spamline = @_;
  my $wordcount= scalar @spamline;  # Number of words in a line of spam
  my $wordlen=0;            # Length of the current word
  my $choice=0;             # Choice of various gene options
  my $length=0;
  my @chromosome=();
  foreach(@spamline){
    s/([\{\}\[\]\(\)\^\$\.\|\*\+\?\\])/\\$1/g;  # Escape all metacharacters
    $wordlen=length($_);
    $choice=int(rand(3));
    if ($choice == 0){
      push @chromosome, ".*";
    }
    #elsif ($choice < ($wordcount/4)){
    elsif ($choice==1){
      push @chromosome, "$_";          # Leave gene unchanged
    }
    else{
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
      if ($wordlen>1){        # If gene is a word (>1 char)
        if (/^(.)\1{2,}$/) {     # Repeated character
          $length=length($_);
          push @chromosome,"$_{$length}";
        }
        elsif (/^\d+$/){         # Numbers
          push @chromosome,"\\d+";   # Numbers
        }
        elsif (/^[A-F0-9]+$/){   # Hex with caps
          push @chromosome,"[A-F0-9]{$wordlen}";
        }
        elsif (/^[a-f0-9]+$/){   # Hex with lowercase
          push @chromosome,"[a-f0-9]{$wordlen}";
        }
        elsif (/^(?:com|net|org|edu|biz|info|us)$/){      # lowercase letters
          push @chromosome,"(?:com|net|org|edu|biz|info|us)";
        }
        elsif (/^[a-z]+$/){      # lowercase letters
          push @chromosome,"[a-z]+";
        }
        elsif (/^[A-Z]+$/){      # Capital letters
          push @chromosome,"[A-Z]+";
        }
        elsif (/^(?:Mon|Tue|Wed|Thu|Fri|Sat|Sun)$/){ # Days of the week
          push @chromosome,"(?:Mon|Tue|Wed|Thu|Fri|Sat|Sun)";
        }
        elsif (/^(?:Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)$/){  # Months
of the year
          push @chromosome,"(?:Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)";
        }
        elsif (/^[A-Z][a-z]+$/){      # Words With The First Letter Capitalized
          push @chromosome,"[A-Z][a-z]+";
        }
        elsif (/^(US)?\$[0-9,]+/){     # Cash amounts such as US$9,300,000
          #print "@spamline\n";
          push @chromosome,"(US)?\$[0-9,]+";
          #print "@chromosome\n";
        }
        else{
          push @chromosome,"$_";
        }
      }
      else{  # Token is 1 character
        if (/[0-9]/){
          push @chromosome,"[0-9]";
        }
        elsif (/[A-Z]/){
          push @chromosome,"[A-Z]";
        }
        elsif (/[a-z]/){
          push @chromosome,"[a-z]";
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
        }
        else {
          push @chromosome,"$_";
        }
      }
    }
  }
  return(@chromosome);
}

sub misses{

  my $hamref=shift;
  my @hamlines=@{$hamref};
  my $regex;
  my $compiled="";  #  compiled regex
  while ( $regex = each %survivorhash ) {
    $compiled= qr/$regex/;  # Much more efficient to compile the regex in advance
    foreach(@hamlines){
      if (!($hamhit{$regex})){
        if (/$compiled/){
          $comparisons++;
          $hamhit{$regex}=1;
          #print "MISS: $regex\n";
        }
      }
    }
  }
  #return($comparisons);
}

sub hits {

  my $spamref=shift;
  my @spamlines=@{$spamref};
  my $compiled="";  #  compiled regex
  my %regexes;
  my $regex;

  while ( $regex = each %survivorhash ) {
    if (!(($regexes{$regex})||($hamhit{$regex}))){  # If it's a new regex (no
previous hits or misses)
      $compiled= qr/$regex/;
      foreach(@spamlines){
        if (/$compiled/){
          $comparisons++;
          $regexes{$regex}++;
          #print "HIT:  $regexes{$regex} $regex\n";
        }
      }
    }
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
  }
  return(%regexes);
}

sub readcorpus {
  # Read the spam and ham corpora from text files
  @hamlines=shift(@_);
  @spamlines=shift(@_);

  #open(HAMTXT,"<ham-bodies0.txt")||die;
  open(HAMTXT,"<ham.txt")||die;
  while(<HAMTXT>){
    push @hamlines,$_ if (/\w/);  # Save lines containing at least 1 word character
  }
  close(HAMTXT);

  #open(SPAMTXT,"<419-bodies0.txt")||die;
  open(SPAMTXT,"<419.txt")||die;
  while(<SPAMTXT>){
    push @spamlines,$_ if (/\w/);  # Save lines containing at least 1 word
character
  }
  close(SPAMTXT);

  #return(@hamlines,@spamlines);
}

sub create_chromosomes{
  my $cref;
  my @currentlines;    # temp array containing spam text which may be spliced, etc.
Needed to leave @spamlines intact
  my @currentchromosomes;   # AoA containing the current chromosomes in play
  my $offset=shift (@_);
  my $length=shift (@_);
  my $survivorref=shift (@_);
  # Seed the first generation
  @currentlines = @spamlines[$offset..$length];  # Grab a chunk of spam
  @currentchromosomes=genregex(\@currentlines); # Generate regexes based on the
spam
  for $cref ( @currentchromosomes ) {
    push @{$survivorref}, [ @$cref ];      # Add the current chromosomes to the
survivor array
  }
}
```

## Appendix C, 'genregex.pl' output

```
# ./genregex.pl
Generation 0 #######################
5.64 /^[A-Z][a-z]+\s+Kingdom\./
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
3.13 /^[a-z]+\s+to\s+[A-Z][a-z]+\s+[a-z]+\s+as\s+Consignment\s+[a-z]+\s+.*[A-
Z]+\s+COURIER\s+/
1.98 /^ive/
1.98 /^UK\./
1.96 /Adjou\./
1.94 /Coulibaly\./
1.93 /^c\.Private\s+/
1.93 /^Dear\s+.*One\./
1.93 /[A-Z]+\s+CENTER/
1.93 /^London,England/
Generation 1 #######################
7.82 /Kingdom\./
7.74 /the\s+finance/
7.34 /(?:^ive|^[A-Z][a-z]+\s+Kingdom\.)/
7.12 /(?:^London,England|^[A-Z][a-z]+\s+Kingdom\.)/
6.76 /(?:^DR\.\s+[A-Z][a-z]+\s+[A-Z][a-z]+|^[A-Z][a-z]+\s+Kingdom\.)/
6.46 /(?:^(?:[a-z]+\s+){2}this\s+transaction,\s+CORPORATE|^[A-Z][a-
z]+\s+Kingdom\.)/
5.88 /^deposit/
5.64 /^[A-Z][a-z]+\s+Kingdom\./
4.08 /(?:^[a-z]+\s+to\s+[A-Z][a-z]+\s+[a-z]+\s+as\s+Consignment\s+[a-z]+\s+.*[A-
Z]+\s+COURIER\s+|he\s+opened\s+[a-z]+\s+.*anonymous\.)/
3.99 /(?:all\s+our\s+.*[A-Z][a-z]+\s+.*categories\.|^[a-z]+\s+to\s+[A-Z][a-
z]+\s+[a-z]+\s+as\s+Consignment\s+[a-z]+\s+.*[A-Z]+\s+COURIER\s+)/
Generation 2 #######################
9.93 /(?:Kingdom\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+)/
9.57 /(?:or\s+his\s+.*of\s+kin\.|^account\s+(?:[a-
z]+\s+){2}.*away,\s+an|the\s+finance)/
9.47 /(?:Adjou\.|Kingdom\.)/
9.38 /(?:Diara\.|the\s+finance)/
8.88 /(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit)/
8.28 /(?:^[A-Z][a-z]+\s+Abdullah\s+Al\s+Futtaim|Adjou\.|^(?:[a-
z]+\s+){2}this\s+transaction,\s+CORPORATE|^[A-Z][a-z]+\s+Kingdom\.)/
7.82 /Kingdom\./
7.80 /Lottery\s+/
7.74 /the\s+finance/
7.66 /(?:^ive|^deposit)/
Generation 3 #######################
25.72 /(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
15.85
/(?:Kingdom\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+|^ive|^deposit
)/
15.35 /(?:the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit)/
15.35 /(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance)/
13.80 /(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit)/
13.16 /(?:^ive|^[A-Z][a-z]+\s+Kingdom\.|MANUEL|^MISS\s+|LORITA\s+|Coulibaly\.)/
12.69 /(?:or\s+his\s+.*of\s+kin\.|^account\s+(?:[a-
z]+\s+){2}.*away,\s+an|the\s+finance|^c\.Private\s+|^London,England|^UK\.)/
11.08 /(?:^vi\.\s+[A-Z][a-z]+|^c\.Private\s+|^[A-Z][a-
z]+\s+Abdullah\s+Al\s+Futtaim|Adjou\.|MANUEL|^MISS\s+|LORITA\s+|Coulibaly\.)/
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
10.78 /(?:^(?:[A-Z]+\s+){2}.*DEPARTMENT|^DR\.\s+[A-Z][a-z]+\s+[A-Z][a-z]+|^[A-Z][a-
z]+\s+Kingdom\.)/
10.64
/(?:MANUEL|^MISS\s+|LORITA\s+|Coulibaly\.|BANK\s+OF\s+NIGERIA\.|Coulibaly\.|LOTTERY
\s+WINNING\s+)/
Generation 4 ######################
27.65
/(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNI
NG\s+|Coulibaly\.|^deposit)/
25.72 /(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
19.04
/(?:Kingdom\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+|^ive|^deposit
|BANK\s+OF\s+NIGERIA\.|Coulibaly\.|LOTTERY\s+WINNING\s+|MANUEL|^MISS\s+|LORITA\s+|C
oulibaly\.)/
16.17
/(?:Kingdom\.|MANUEL|^MISS\s+|LORITA\s+|Coulibaly\.|BANK\s+OF\s+NIGERIA\.|Coulibaly
\.|LOTTERY\s+WINNING\s+)/
15.85
/(?:Kingdom\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+|^ive|^deposit
)/
15.37 /(?:or\s+his\s+.*of\s+kin\.|^account\s+(?:[a-
z]+\s+){2}.*away,\s+an|the\s+finance|^c\.Private\s+|^London,England|^UK\.|Coulibaly
\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+|^deposit)/
15.35 /(?:the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit)/
15.35 /(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance)/
14.77
/(?:Kingdom\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+|^vi\.\s+[A-
Z][a-z]+|^c\.Private\s+|^[A-Z][a-
z]+\s+Abdullah\s+Al\s+Futtaim|Adjou\.|MANUEL|^MISS\s+|LORITA\s+|Coulibaly\.)/
14.52 /(?:or\s+his\s+.*of\s+kin\.|^account\s+(?:[a-
z]+\s+){2}.*away,\s+an|the\s+finance|Coulibaly\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\
s+Box\s+.*Station,\s+|^deposit)/
Generation 5 ######################
29.64
/(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit)/
27.82
/(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit|or\s+his\s+.*of\s+kin\.|^
account\s+(?:[a-z]+\s+){2}.*away,\s+an|the\s+finance)/
27.65
/(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNI
NG\s+|Coulibaly\.|^deposit)/
26.62 /(?:or\s+his\s+.*of\s+kin\.|^account\s+(?:[a-
z]+\s+){2}.*away,\s+an|the\s+finance|MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^de
posit|LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
25.72 /(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
23.50
/(?:Kingdom\.|MANUEL|^MISS\s+|LORITA\s+|Coulibaly\.|BANK\s+OF\s+NIGERIA\.|Coulibaly
\.|LOTTERY\s+WINNING\s+|MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY
\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
22.22
/(?:Kingdom\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+|^ive|^deposit
|BANK\s+OF\s+NIGERIA\.|Coulibaly\.|LOTTERY\s+WINNING\s+|MANUEL|^MISS\s+|LORITA\s+|C
oulibaly\.|Diara\.|the\s+finance)/
19.60
/(?:Adjou\.|Kingdom\.|the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|or\s+
his\s+.*of\s+kin\.|^account\s+(?:[a-z]+\s+){2}.*away,\s+an|the\s+finance)/
19.04
/(?:Kingdom\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+|^ive|^deposit
|BANK\s+OF\s+NIGERIA\.|Coulibaly\.|LOTTERY\s+WINNING\s+|MANUEL|^MISS\s+|LORITA\s+|C
oulibaly\.)/
18.90
/(?:Kingdom\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+|^vi\.\s+[A-
Z][a-z]+|^c\.Private\s+|^[A-Z][a-
z]+\s+Abdullah\s+Al\s+Futtaim|Adjou\.|MANUEL|^MISS\s+|LORITA\s+|Coulibaly\.|[a-
z]+\s+.*response\.)/
Generation 6 ######################
29.64
/(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit)/
29.44
/(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit|the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit)/
27.82
/(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit|or\s+his\s+.*of\s+kin\.|^
account\s+(?:[a-z]+\s+){2}.*away,\s+an|the\s+finance)/
27.65
/(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNI
NG\s+|Coulibaly\.|^deposit)/
27.27
/(?:the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|LOTTERY\s+|Lottery\s+|W
INNING\s+|Coulibaly\.|^deposit)/
26.62 /(?:or\s+his\s+.*of\s+kin\.|^account\s+(?:[a-
z]+\s+){2}.*away,\s+an|the\s+finance|MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^de
posit|LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
25.72 /(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
24.61
/(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance|Coulibaly\.|LOTTERY\s+W
INNING\s+|^deposit|^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|
Coulibaly\.|^deposit)/
23.50
/(?:Kingdom\.|MANUEL|^MISS\s+|LORITA\s+|Coulibaly\.|BANK\s+OF\s+NIGERIA\.|Coulibaly
\.|LOTTERY\s+WINNING\s+|MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY
\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
23.12
/(?:Kingdom\.|MANUEL|^MISS\s+|LORITA\s+|Coulibaly\.|BANK\s+OF\s+NIGERIA\.|Coulibaly
\.|LOTTERY\s+WINNING\s+|[a-z]+\s+.*response\.)/
Generation 7 ######################
30.24 /YOUR\s+[A-Z]+\s+[A-Z]+/
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
29.64
/(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit)/
29.44
/(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit|the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit)/
27.82
/(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit|or\s+his\s+.*of\s+kin\.|^
account\s+(?:[a-z]+\s+){2}.*away,\s+an|the\s+finance)/
27.65
/(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNI
NG\s+|Coulibaly\.|^deposit)/
27.27
/(?:the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|LOTTERY\s+|Lottery\s+|W
INNING\s+|Coulibaly\.|^deposit)/
27.03
/(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNI
NG\s+|Coulibaly\.|^deposit|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance)
/
26.62 /(?:or\s+his\s+.*of\s+kin\.|^account\s+(?:[a-
z]+\s+){2}.*away,\s+an|the\s+finance|MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^de
posit|LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
25.72 /(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)/
24.61
/(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance|Coulibaly\.|LOTTERY\s+W
INNING\s+|^deposit|^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|
Coulibaly\.|^deposit)/
Generation 8 ######################
39.75 /(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance|YOUR\s+[A-
Z]+\s+[A-Z]+)/
31.17 /(?:YOUR\s+[A-Z]+\s+[A-Z]+|^ive|^[A-Z][a-
z]+\s+Kingdom\.|Coulibaly\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+
|^deposit|MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit)/
30.24 /YOUR\s+[A-Z]+\s+[A-Z]+/
29.64
/(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit)/
29.44
/(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit|the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit)/
27.82
/(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit|or\s+his\s+.*of\s+kin\.|^
account\s+(?:[a-z]+\s+){2}.*away,\s+an|the\s+finance)/
27.65
/(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNI
NG\s+|Coulibaly\.|^deposit)/
27.27
/(?:the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|LOTTERY\s+|Lottery\s+|W
INNING\s+|Coulibaly\.|^deposit)/
27.03
/(?:the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|MANUEL|^MISS\s+|^ive|LO
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

```
RITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)
/
27.03
/(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNI
NG\s+|Coulibaly\.|^deposit|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance)
/
Generation 9 ######################
50.58 /(?:YOUR\s+[A-Z]+\s+[A-
Z]+|^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit)/
39.75 /(?:Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|the\s+finance|YOUR\s+[A-
Z]+\s+[A-Z]+)/
31.17 /(?:YOUR\s+[A-Z]+\s+[A-Z]+|^ive|^[A-Z][a-
z]+\s+Kingdom\.|Coulibaly\.|Johnson\s+Mba\.|^PlasmaNet\s+.*PO\s+Box\s+.*Station,\s+
|^deposit|MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit)/
30.24 /YOUR\s+[A-Z]+\s+[A-Z]+/
29.64
/(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit)/
29.44
/(?:^MISS\s+|^ive|MANUEL|LOTTERY\s+|Lottery\s+|WINNING\s+|LORITA\s+|Coulibaly\.|^de
posit|the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit)/
27.82
/(?:LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit|or\s+his\s+.*of\s+kin\.|^
account\s+(?:[a-z]+\s+){2}.*away,\s+an|the\s+finance)/
27.65
/(?:MANUEL|^MISS\s+|^ive|LORITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNI
NG\s+|Coulibaly\.|^deposit)/
27.27
/(?:the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|LOTTERY\s+|Lottery\s+|W
INNING\s+|Coulibaly\.|^deposit)/
27.03
/(?:the\s+finance|Coulibaly\.|LOTTERY\s+WINNING\s+|^deposit|MANUEL|^MISS\s+|^ive|LO
RITA\s+|Coulibaly\.|^deposit|LOTTERY\s+|Lottery\s+|WINNING\s+|Coulibaly\.|^deposit)
/
Comparisons: 8088.  Comparisons/sec:38.8846153846154. Time taken was 208 seconds
```



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

# 9. References

## Documents

Dalton, John. *Newcastle Engineering Design Centre, January 2007: Genetic Algorithms (GAs)*. Retrieved November 1, 2007, from http://www.edc.ncl.ac.uk/highlight/rhjanuary2007.php

Dracopoulos, Dimitris C. *2AIT608 - Machine Learning Genetic Algorithms.* Retrieved November 4, 2007, from http://users.wmin.ac.uk/~dracopd/DOCUM/courses/2ait608/genetic_algorithms_lecture_notes.pdf

The Federal Bureau of Investigation. *Common Fraud Schemes.* Retrieved September 26, 2007, from http://www.fbi.gov/majcases/fraud/fraudschemes.htm

Goldberg, David E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

Graham, P. (August 2002). *A Plan for Spam.* Retrieved September 26, 2007, from http://www.paulgraham.com/spam.html

Holland, John H. "Genetic Algorithms." *Scientific American*. July 1992. URL: http://www.econ.iastate.edu/tesfatsi/holland.GAIntro.htm

The International Centre for Nigerian Law. *Nigerian Criminal Code Act-Part VI to the end.* Retrieved September 26, 2007, from http://www.nigeria-law.org/Criminal%20Code%20Act-Part%20VI%20%20to%20the%20end.htm

Joyce, J. (2007). Bayes' Theorem, *The Stanford Encyclopedia of Philosophy (Summer 2007 Edition)*, Edward N. Zalta (ed.), Retrieved September 26, 2007, from http://plato.stanford.edu/archives/sum2007/entries/bayes-theorem/

Wong, H. (1996). Introduction to Genetic Algorithms. *Surveys and Presentations in Information Systems Engineering (SURPRISE) Journal 96 Volume 4.* Retrieved September 26, 2007, from http://www-dse.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

## Tools

GNU sort. URL: http://www.gnu.org/software/coreutils/

Perl. URL: http://www.perl.org/

Perl Compatible Regular Expressions. URL: http://www.pcre.org/

Postfix. URL: http://www.postfix.org/

SpamAssassin. URL: http://spamassassin.apache.org

© SANS Institute 2007, Author retains full rights.



© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Last Updated: May 25th, 2013

# Upcoming SANS Training

**Click Here for a full list of all Upcoming SANS Events by Location**

| | | | |
|---|---|---|---|
| SANS Malaysia @ MCMC 2013 | Cyberjaya, MY | Jun 03, 2013 - Jun 08, 2013 | Live Event |
| SANS Pen Test Berlin 2013 | Berlin, DE | Jun 03, 2013 - Jun 08, 2013 | Live Event |
| Industial Control Systems Security Training - Houston | Houston, TXUS | Jun 10, 2013 - Jun 15, 2013 | Live Event |
| Security Impact of IPv6 Summit 2013 | Washington, DCUS | Jun 14, 2013 - Jun 16, 2013 | Live Event |
| SANSFIRE 2013 | Washington, DCUS | Jun 14, 2013 - Jun 22, 2013 | Live Event |
| SANS Canberra 2013 | Canberra, AU | Jul 01, 2013 - Jul 13, 2013 | Live Event |
| Digital Forensics & Incident Response Summit 2013 | Austin, TXUS | Jul 09, 2013 - Jul 16, 2013 | Live Event |
| SANS London Summer 2013 | London, GB | Jul 09, 2013 - Jul 16, 2013 | Live Event |
| SANS Rocky Mountain 2013 | Denver, COUS | Jul 14, 2013 - Jul 20, 2013 | Live Event |
| SANS Mumbai 2013 | Mumbai, IN | Jul 22, 2013 - Jul 27, 2013 | Live Event |
| SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification | Washington, DCUS | Jul 22, 2013 - Jul 26, 2013 | Live Event |
| SANS San Francisco 2013 | San Francisco, CAUS | Jul 29, 2013 - Aug 03, 2013 | Live Event |
| SANS SEC 560: Network Penetration Testing @ Bangalore 2013 | Bangalore, IN | Aug 05, 2013 - Aug 10, 2013 | Live Event |
| SANS Boston 2013 | Boston, MAUS | Aug 05, 2013 - Aug 10, 2013 | Live Event |
| Critical Security Controls Summit | Washington, DCUS | Aug 12, 2013 - Aug 18, 2013 | Live Event |
| Industrial Control Systems Security Training - DC | Washington, DCUS | Aug 12, 2013 - Aug 16, 2013 | Live Event |
| SANS Thailand 2013 | Bangkok, TH | Aug 19, 2013 - Aug 31, 2013 | Live Event |
| SANS Virginia Beach 2013 | Virginia Beach, VAUS | Aug 19, 2013 - Aug 30, 2013 | Live Event |
| Mobile Device Security Summit 2013 | OnlineCAUS | May 30, 2013 - Jun 06, 2013 | Live Event |
| SANS OnDemand | Books & MP3s OnlyUS | Anytime | Self Paced |