Roll your dice and take your chances

Thiébaud Weksteen
<thiebaud@weksteen.fr>

Introduction

- Why this talk?
- Tokens?
- Practical Approach
- Introduction to statistical testing
- What's next?

Why this talk ?

- “I've checked the session tokens, they are random”
- “These IDs are not random, the graphs are red”
- “The key space seems large enough”
- ...

Tokens » Web Examples

- Classic
  - Session tokens (JSESSIONID, PHPSESSID)
  - CSRF tokens
  - CAPTCHA
- Application tokens
  - Order reference number
  - Advertisement ID
  - File ID

Tokens » Real Life Examples

- Security Tokens
  - RSA token generator
  - Symantec VIP
- Misc
  - Flight Booking Number
  - Coupon Code

Tokens » Real Life Examples

- Tamworth Library Internet Access ticket

Tokens » Definition

- Definition: unique identifier
- Attributes:
  - Lifetime
  - Bound to other entities
  - Space
- Usage:
  - Reference an entity (user, file, device, ...)
  - Authenticate/Authorise/Validate origin (CSRF, CAPTCHA)

Tokens » Generation

- Random number generator
- Other identifiers (global/local)
  - Database/LDAP ids
  - Application Server id
  - Memory address
  - Serial Number
  - IMEI
- Time
  - UNIX epoch (`date +%s` ~ 1379636110)
  - Time Zone
- Constant

Tokens » Generation Examples

- RSA Software Serial Number [1]

  Used to bind a software token to a device

```
SHA1(host_name + user_SID + “RSA Copyright 2008”) << 10
```

Tokens » Generation Examples

- Rails
  - Session

```
def generate_sid
  sid = SecureRandom.hex(16)
  sid.encode!(Encoding::UTF_8)
  sid
end
```

  - CSRF protection

```
def form_authenticity_token
  session[:_csrf_token] ||= SecureRandom.base64(32)
end
```

Tokens » Fail Example

- CVE-2010-4568
  - “Bugzilla before 4.0rc2 does not properly generate random values for cookies and tokens, which allows remote attackers to obtain access to arbitrary accounts via unspecified vectors, related to an insufficient number of calls to the srand function.”
- Random generator state shared
- Limited key space (32 bits)

Tokens » Attacks

- Enumeration:
  - Access unauthorised data (if no extra access control)
    E.g., user accessing other user's file
  - May require brute force, how much?
- Collision
- Space reduction
- Fun

Token Testing » Introduction

- Goals
  - Can I find already existing tokens?
    - Part or whole?
    - How predictable?
  - Can I predict which token will be emitted under certain circumstances?
    - What's the impact of unknown parameters?

Token Testing » Our Focus

- Black-box approach
  - Algorithm unknown
  - Partial knowledge on some parameters
- Exact alphabet unknown
  - Binary approach
  - Character approach
- Limited sample size
- Reason for lack of randomness matters

Token Testing » Practical Approach

```
1000$D8O-_z02dUbOAD6sSVvxANk:13l5t867f

1000$oMjRxMc5rqVZMRedRI_LctT:13l5t867f

1000$dX4Dk5rOAEl2CCUJNXsQClz:13l5t867f

1000$4FQaj10lOZQeu3clVsZc04p:13l5t86cq

1000$poGpeogXmlf_OMTYojzmgYm:13l5t867f
```

Token Testing » Practical Approach

```
1000$D8O-_z02dUbOAD6sSVvxANk:13l5t867f

1000$oMjRxMc5rqVZMRedRI_LctT:13l5t867f

1000$dX4Dk5rOAEl2CCUJNXsQClz:13l5t867f

1000$4FQaj10lOZQeu3clVsZc04p:13l5t86cq

1000$poGpeogXmlf_OMTYojzmgYm:13l5t867f
```

Fixed length

Token Testing » Practical Approach

```
1000$D8O-_z02dUbOAD6sSVvxANk:13l5t867f
1000$oMjRxMc5rqVZMRedRI_LctT:13l5t867f
1000$dX4Dk5rOAEl2CCUJNXsQClz:13l5t867f
1000$4FQaj10lOZQeu3clVsZc04p:13l5t86cq
1000$poGpeogXmlf_OMTYojzmgYm:13l5t867f
```

Delimiters?

Token Testing » Practical Approach

```
1000$D8O-_z02dUbOAD6sSVvxANk:13l5t867f
1000$oMjRxMc5rqVZMRedRI_LctT:13l5t867f
1000$dX4Dk5rOAEl2CCUJNXsQClz:13l5t867f
1000$4FQaj10lOZQeu3clVsZc04p:13l5t86cq
1000$poGpeogXmlf_OMTYojzmgYm:13l5t867f
```

Decimal? Base64 (urlsafe)? [a-z0-9]?

Token Testing » Practical Approach

```
1000$D8O-_z02dUbOAD6sSVvxANk:13l5t867f
1000$oMjRxMc5rqVZMRedRI_LctT:13l5t867f
1000$dX4Dk5rOAEl2CCUJNXsQClz:13l5t867f
1000$4FQaj10lOZQeu3clVsZc04p:13l5t86cq
1000$poGpeogXmlf_OMTYojzmgYm:13l5t867f
```

Constant ? Alternating?

Token Testing » Practical Approach

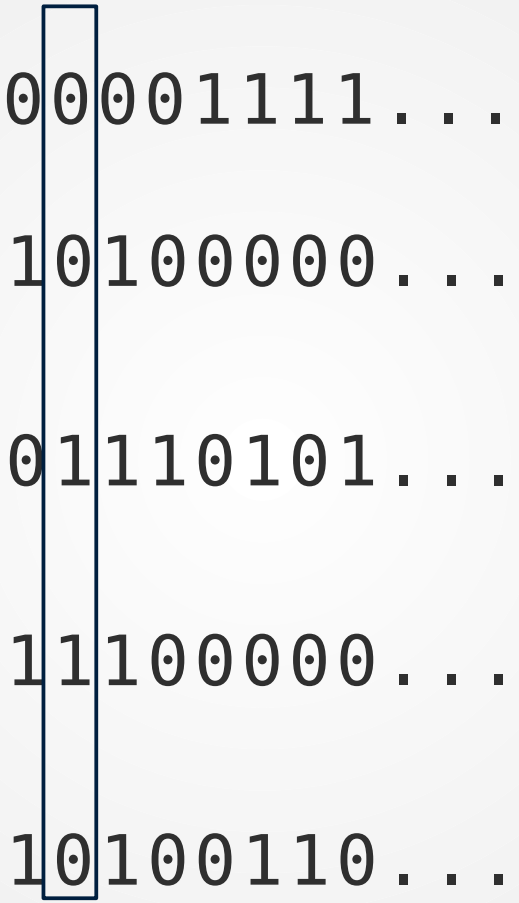
```
D8O-_z02dUbOAD6sSVvxANk

oMjRxMc5rqVZMRedRI_LctT

dX4Dk5rOAEl2CCUJNXsQClz

4FQaj10lOZQeu3clVsZc04p

poGpeogXmlf_OMTYojzmgYm
```

Token Testing » Practical Approach

Token Testing » Practical Approach

```
00001111...

10100000...

01110101...

11100000...

10100110...
```

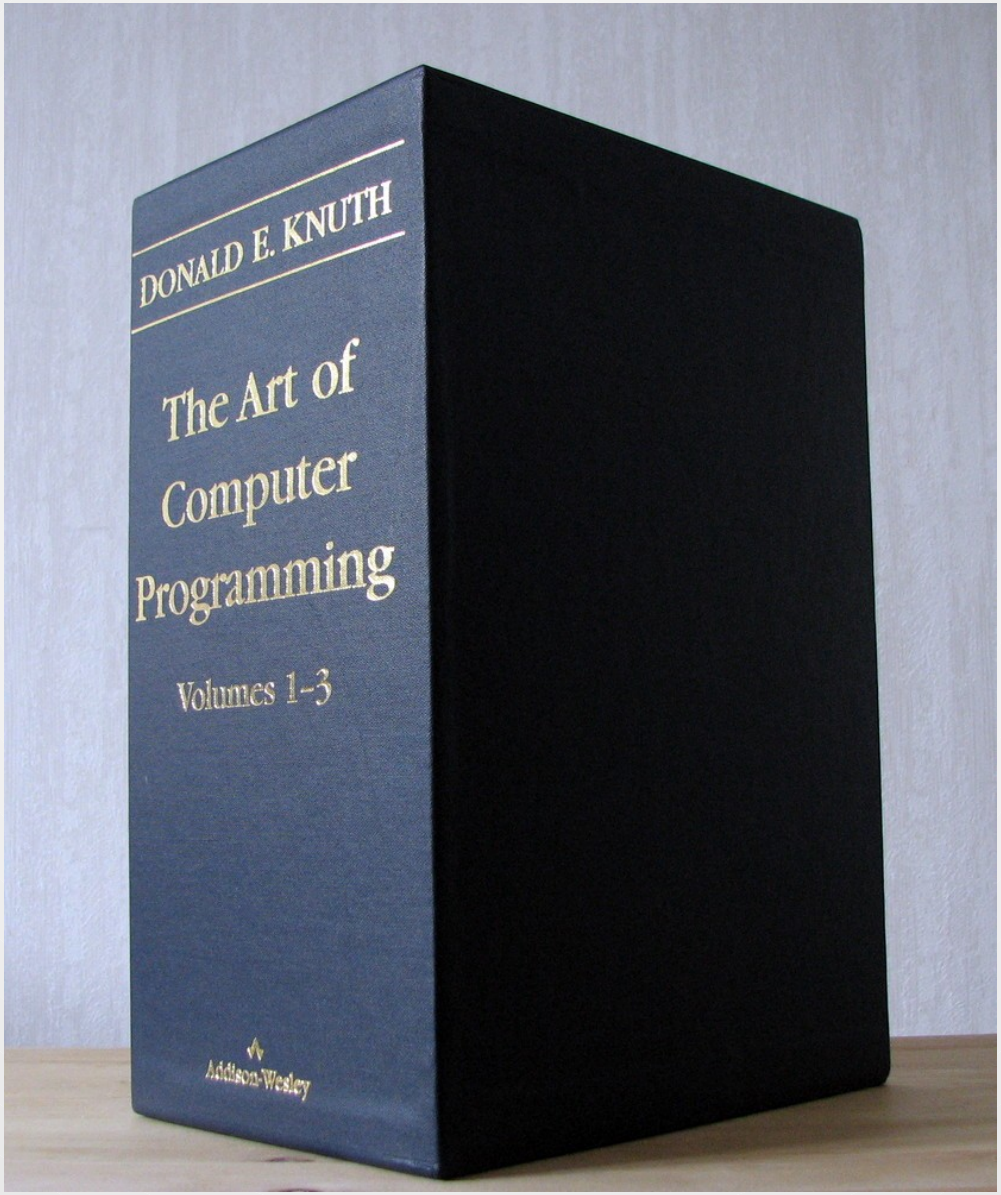
Token Testing » Practical Approach



Token Testing » Practical Approach

- Use Statistical Hypothesis Testing from FIPS 140-1
  - Assume it is randomly generated
  - Measure some statistic
  - Decide if hypothesis holds
- Available Tools:
  - Burp Sequencer
  - Stompy (by lcamtuf)

Statistical Testing » History



[2]

Interlude

- The Art of Computer Programming. (1981) Vol 2 Chapter 3 ends with...

  “Every reader is urged to work Exercice 6 in the following set of problems”

  Exercice 6: “Look at the subroutine library of each computer installation in your organisation, and replace the random number generators by good ones. Try to avoid being too shocked by what you find.”

Statistical Testing » History

Computers Math. Applic. Vol. 26, No. 9, pp. 1–10, 1993
Printed in Great Britain.

0898-1221/93 $6.00 + 0.00

Monkey Tests for Random Number Generators

G. MARSAGLIA AND A. ZAMAN
Supercomputer Computations Research Institute
and
Department of Statistics, The Florida State University
Tallahasse, FL 32306, U.S.A.

(Received and accepted May 1993)

Abstract—This article describes some very simple, as well as some quite sophisticated, tests that shed light on the suitability of certain random number generators.

INTRODUCTION

Few images invoke the mysteries and ultimate certainties of a sequence of random events as well as that of the proverbial monkey at a typewriter. Surprisingly, many questions about the monkey's literary output—the times between appearances of certain strings, the number of distinct four-

Statistical Testing » History

www.stat.fsu.edu/pub/diehard/

The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness

THE MARSAGLIA
RANDOM NUMBER CDROM
including the
DIEHARD BATTERY OF TESTS
OF RANDOMNESS

Research
Sponsored by
THE NATIONAL
SCIENCE
FOUNDATION
Grants
DMS-8807976
DMS-9206972

DEPARTMENT
OF STATISTICS
and
SUPERCOMPUTER
COMPUTATIONS
RESEARCH
INSTITUTE

Professor
George Marsaglia
geo@stat.fsu.edu

Research Sponsored by the National Science Foundation (Grants DMS-8807976 and DMS-9206972), copyright 1995 George Marsaglia.

Browse the contents of the CD-ROM!

Statistical Testing » History

Random Number Generators and Empirical Tests

Pierre L'Ecuyer

ABSTRACT We recall some requirements for "good" random number generators and argue that while the construction of generators and the choice of their parameters must be based on theory, a posteriori empirical testing is also important. We then give examples of tests failed by some popular generators and examples of generators passing these tests.

1 Introduction

Roughly, a random number generator is a simple program producing a periodic sequence of numbers on a computer. This sequence should behave,

Statistical Testing » History

FIPS PUB 140-2

CHANGE NOTICES (12-03-2002)

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION
(Supercedes FIPS PUB 140-1, 1994 January 11)

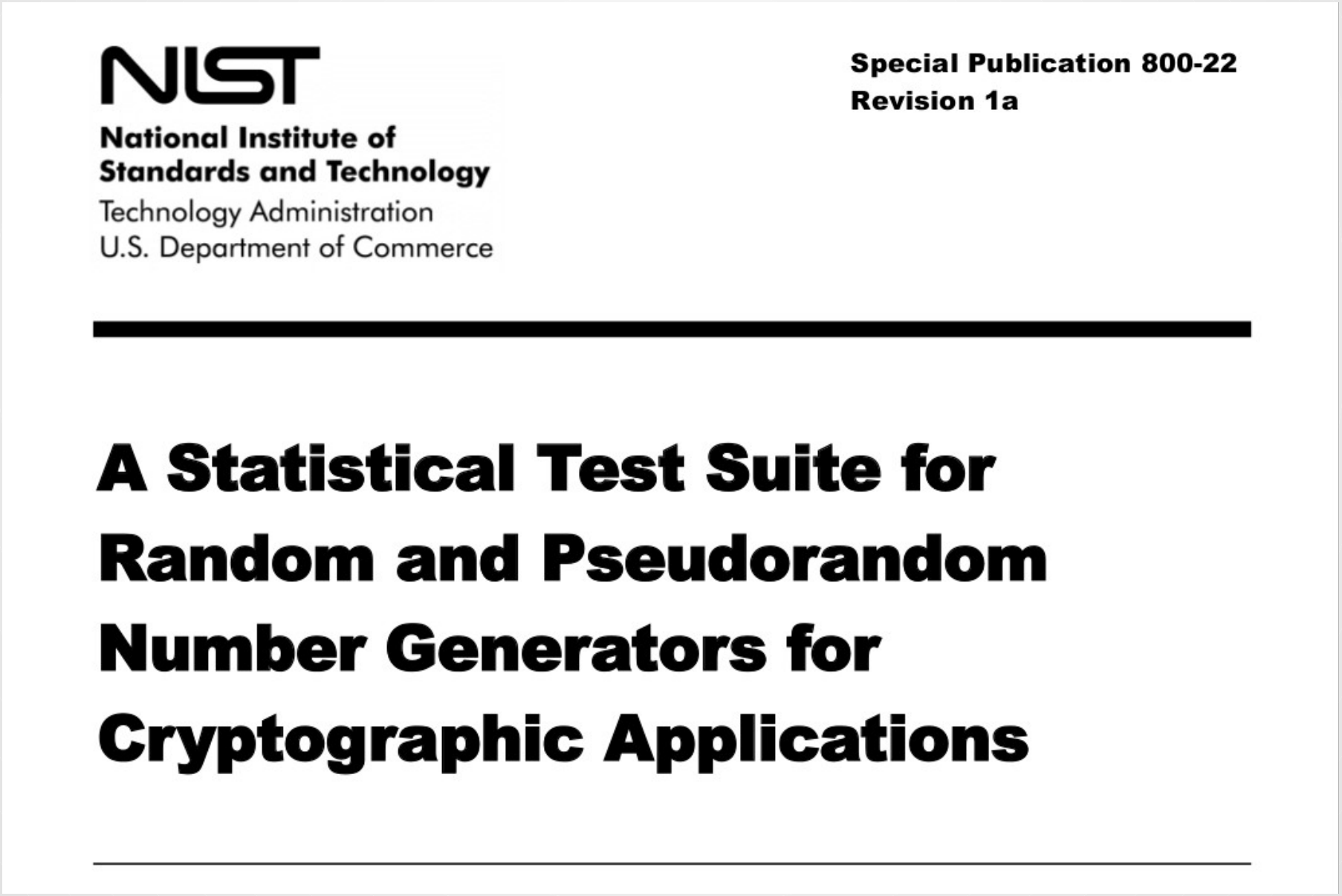
SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES

CATEGORY: COMPUTER SECURITY **SUBCATEGORY: CRYPTOGRAPHY**

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

Issued May 25, 2001

Statistical Testing » History



Statistical Testing » History

Corrections of the NIST Statistical Test Suite for Randomness

Song-Ju Kim, Ken Umeno, and Akio Hasegawa

Chaos-based Cipher Chip Project, Presidential Research Fund,
Communications Research Laboratory, Incorporated Administrative Agency
4-2-1, Nukui-kitamachi, Koganei-shi, Tokyo 184-8795, Japan
{songju, umeno, ahase}@crl.go.jp

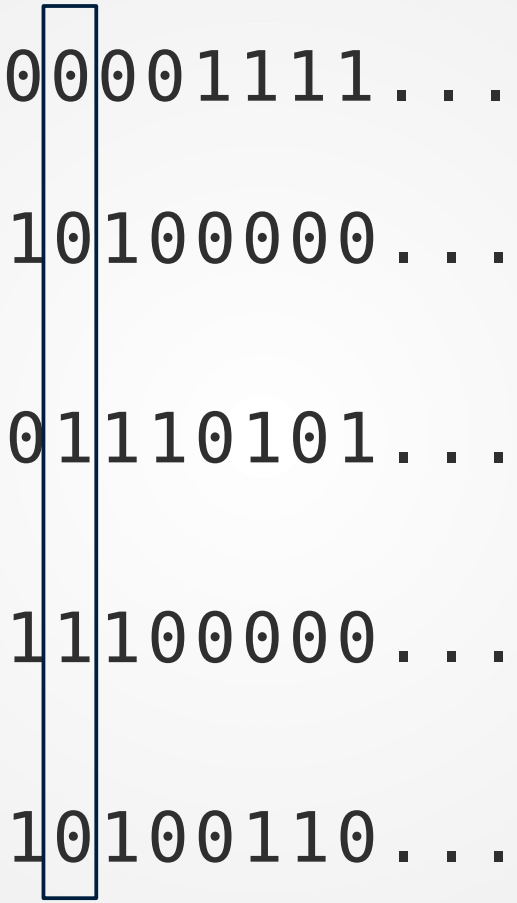
Abstract. It is well known that the NIST statistical test suite was used for the evaluation of AES candidate algorithms. We have found that the test setting of Discrete Fourier Transform test and Lempel-Ziv test of this test suite are wrong. We give four corrections of mistakes in the test settings. This suggests that re-evaluation of the test results should be needed.

Key words: Pseudo-Random Bit Generator, Statistical Test, Discrete Fourier Transform, Lempel-Ziv Compression Algorithm, Cellular Automata

Statistical Testing » Theory

- Statistical Testing (Fisher/p-value variant)
  - Let $\mathcal{H}_o$ = “sample is randomly generated”
  - Under $\mathcal{H}_o$, a statistic (*s*) based on the sample should follow a certain distribution
  - Using this distribution, calculate a *p-value,* probability that the expected statistic is at least as extreme as the observed statistic
  - Calculate the observed statistic
  - If *p-value* ≤ α, reject $\mathcal{H}_o$

Statistical Testing » Theory



Statistical Testing » Frequency Test

- Idea: Test the proportion of “0” and “1”
- Statistic:
  - If random (Bernoulli trial), the ~sum~ should follow a standard half-normal distribution:

$$S = \frac{|\sum 2\mathrm{b}_i - 1|}{\sqrt{n}}$$

  - P-value: $P(S > S_{obs}) = erfc\left(\frac{S_{obs}}{\sqrt{2}}\right)$

- Variant: larger alphabet, by block (local frequency)

Statistical Testing » Frequency Test

```
  say("=> Analysis done, ready to execute statistical tests.\n\n");

  say("[*] Checking alphabet usage uniformity... ");
  fflush(0);

  for (l=0;l<maxlen[i];l++) {
    _s32 min, max;
    float mid, dev;
    if (alsize[i][l] <= 1) continue;
    mid = 20000.0 / alsize[i][l];
    dev = MDEV(alsize[i][l]);                          #define MDEV(_x) (500 / pow(_x,0.375) + 5)
    min = mid - dev;
    max = mid + dev;
    for (c=0;c<256;c++) {
      if (alphabet[i][l][c] && alphabet[i][l][c] < min) {
        if (reported_flaws < 6) {
          if (!reported_flaws) say("FAILED\n");
          reported_flaws++;
        } else skipped_flaws++;

define say_rep(x...) do { if (reported_flaws < 6) say(x); else say_tofile(x); } while (0)

        say_rep("    Character '%c' is too rare at position #%u (%u, accept min: %d).\n",
                c,l,alphabet[i][l][c],min);
        badbytes[l]++;
        entropy_loss += port_log2(alsize[i][l]) * 1.0 / alsize[i][l];

      } else if (alphabet[i][l][c] && alphabet[i][l][c] > max) {
        if (reported_flaws < 6) {
          if (!reported_flaws) say("FAILED\n");
```

Statistical Testing » Serial Matching

- Idea: Test transition frequency, each transition should appears as often as the others.
- Statistic: (non-overlapping)

$$S = \sum \frac{\left(c_{obs\{i,j\}} - c_{expected}\right)^2}{c_{expected}}$$

$$P\left(S > S_{obs}\right) = \frac{\gamma\left(\frac{k}{2}; \frac{S_{obs}}{2}\right)}{\Gamma\left(\frac{k}{2}\right)}$$

- Variants: Monkey testing, OPSO, OTSO...

Statistical Testing » Run Test

- Idea: Test number uninterrupted sequence of identical bits
- Statistic:

$$S=\sum r(k)+1 \quad r(k)=0 \quad if \quad b_k=b_{k+1} \quad else \quad 0$$

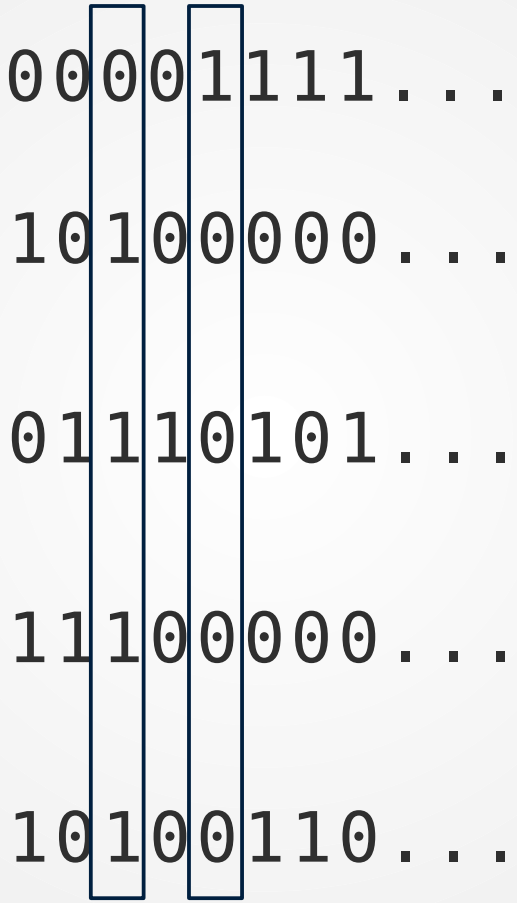
$$P(S>S_{obs})=erfc(\frac{|S_{obs}-\mu|}{\sigma\sqrt{2}})$$

- Variants: by block
- No direct equivalent for characters
- NIST approximation

Statistical Testing » Etc

- Tons other tests exist:
  - Spectral test (DFT)
  - Binary Rank Matrix
  - Linear Complexity
  - Compression / Lempel-Ziv
  - Collision
  - ...

Statistical Testing » Theory



Statistical Testing » Correlation

- Idea: Two sequences should be independants
- Statistic similar to serial matching
- Variant: Sliding window

Statistical Testing » Example

```
271a3780ce5911e2b8bf100077a150e2
274147b2ce5911e2a9ce100077a150e2
27591590ce5911e2b3e8100077a150e2
2770472ece5911e29280100077a150e2
2789e9ccce5911e29fae100077a150e2
27a07f2ace5911e287ce100077a150e2
27b8e948ce5911e2b024100077a150e2
27d6d1b0ce5911e2bb01100077a150e2
27f2e54ece5911e2814f100077a150e2
280709acce5911e28a67100077a150e2
281d02c0ce5911e297ec100077a150e2
2834d09ece5911e2a2ea100077a150e2
284c023cce5911e2b00e100077a150e2
2865089ace5911e2831e100077a150e2
2885fe42ce5911e2b863100077a150e2
```

Statistical Testing » Example

```
Running serial_test_nonoverlap at position 11
  Local Character Set: 9ab
  Character set is tiny
  X^2 = 5172.960800
  serial_test_nonoverlap has failed (character position=11, p-value=0.0)
  Reason:
  Some character transitions are more probable than others:
  Min=0 Max=948 E.Avg=277.777777778
  {'aa': 948
   '99': 895
   'bb': 657
   'ab': 0
   'ba': 0
   '9a': 0
   '9b': 0
   'a9': 0
   'b9': 0}
Running serial_test_nonoverlap at position 12
  Local Character Set: 1
  Character set is tiny
  X^2 = 0.000000
  serial_test_nonoverlap has passed (character position=12, p-value=1.0)
Running serial_test_nonoverlap at position 13
  Local Character Set: 1
  Character set is tiny
  X^2 = 0.000000
  serial_test_nonoverlap has passed (character position=13, p-value=1.0)
Running serial_test_nonoverlap at position 14
  Local Character Set: e
  Character set is tiny
  X^2 = 0.000000
  serial_test_nonoverlap has passed (character position=14, p-value=1.0)
Running serial_test_nonoverlap at position 15
  Local Character Set: 2
  Character set is tiny
  X^2 = 0.000000
  serial_test_nonoverlap has passed (character position=15, p-value=1.0)
Running serial_test_nonoverlap at position 16
  Local Character Set: 89ab
  Character set is tiny
  X^2 = 7.033600
:
```

Statistical Testing » Example

```
271a3780-ce59-11e2-b8bf-100077a150e2
274147b2-ce59-11e2-a9ce-100077a150e2
27591590-ce59-11e2-b3e8-100077a150e2
2770472e-ce59-11e2-9280-100077a150e2
2789e9cc-ce59-11e2-9fae-100077a150e2
27a07f2a-ce59-11e2-87ce-100077a150e2
27b8e948-ce59-11e2-b024-100077a150e2
27d6d1b0-ce59-11e2-bb01-100077a150e2
27f2e54e-ce59-11e2-814f-100077a150e2
280709ac-ce59-11e2-8a67-100077a150e2
```

Statistical Testing » End?

[3]

Token Testing » Practical Approach

```
1000$D8O-_z02dUbOAD6sSVvxANk:13l5t867f

1000$oMjRxMc5rqVZMRedRI_LctT:13l5t867f

1000$dX4Dk5rOAEl2CCUJNXsQClz:13l5t867f

1000$4FQaj10lOZQeu3clVsZc04p:13l5t86cq

1000$poGpeogXmlf_OMTYojzmgYm:13l5t867f
```

What did we miss?

Machine Learning Testing »
Introduction

- Idea:
  - Create *2* sets of measures at significantly different dates
  - Build a set of features (frequency, transition, position, ...)
  - Classify and test
- If successful, the tokens are likely to be based on a time parameter
- Depending on classification used, we know where the difference is coming from

Machine Learning Testing » Theory

- Apply to any attacker-controlled parameter:
  - Time
  - User
  - Global Id
- Parameter dependancy is the biggest issue

Machine Learning Testing » Theory

- Using Logistic Regression
  - Simplest ML algorithm for classification
  - Builds weights vector + Bias
  - Cross-validate to deduce accuracy
  - Using weights: able to find linear combination between features that will lead to a conclusion

Machine Learning Testing » PoC

```
[tweek@sec0 exonumia]$ ./classify.py  ./examples/uuid1.txt ./examples/uuid12.txt
Reading tokens
Alphabet contains 17 characters: -0123456789abcdef
Building features
918 features have been generated
Cross-validating the model
Accuracy: 1.00 (+/- 0.00)
-0.958843562052 a@2
-0.958843562052 a@19
-0.958843562052 0@21
-0.958843562052 4@0
-0.958843562052 6@22
...
0.958859386828 0@12
0.958859386828 9@19
0.958859386828 9@21
0.958859386828 9@22
0.958859386828 8@2
[tweek@sec0 exonumia]$ ./classify.py  ./examples/uuid4.txt ./examples/uuid42.txt
Reading tokens
Alphabet contains 17 characters: -0123456789abcdef
Building features
918 features have been generated
Cross-validating the model
Accuracy: 0.50 (+/- 0.05)
-0.703887294798 #e1
-0.694081569065 #73
-0.687461383703 #6d
-0.651249918489 2@25
```

Things to Remember

- Measure
  - Environment
  - Conditions
- Decode
  - Base16/Base32*/Base64{urlsafe}/hex
  - URL encoded
  - UUID
- Type of analysis

Code

- BSD License
- test.py
- classify.py

```
Running freq_test at position 10
  Local Character Set: 5
  Character set is tiny
  X^2 = 0.000000
  freq_test has passed (character position=10, p-value=1.0)
Running freq_test at position 11
  Local Character Set: 9ab
  Character set is tiny
  X^2 = 115.307200
  freq_test has failed (character position=11, p-value=0.0)
  Reason:
  Some characters appear not or too often. Below is the occurrence r
  Min=1314 Max=1896 E.Avg=1666.66666667
  {'a': 1896
   '9': 1790
   'b': 1314}
Running freq_test at position 12
  Local Character Set: 1
  Character set is tiny
  X^2 = 0.000000
  freq_test has passed (character position=12, p-value=1.0)
```

Questions?

Thanks for your attention!

`github.com/tweksteen/exonumia`

References

- [1] http://www.sensepost.com/blog/7045.html
- [2] http://coderaptors.com/?TAoCP
- [3] http://www.flickr.com/photos/8510057@N02