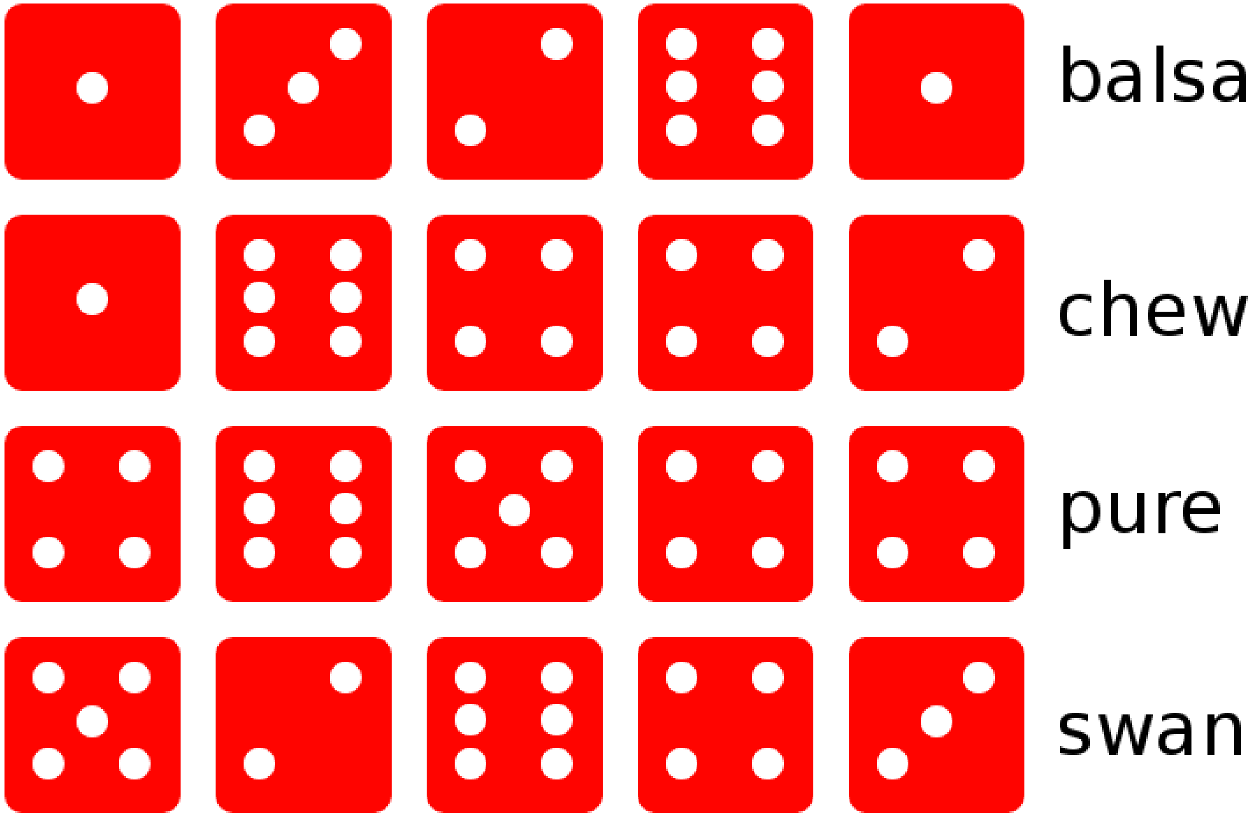
Probabilistic Passphrase Cracking

Luc Gommans

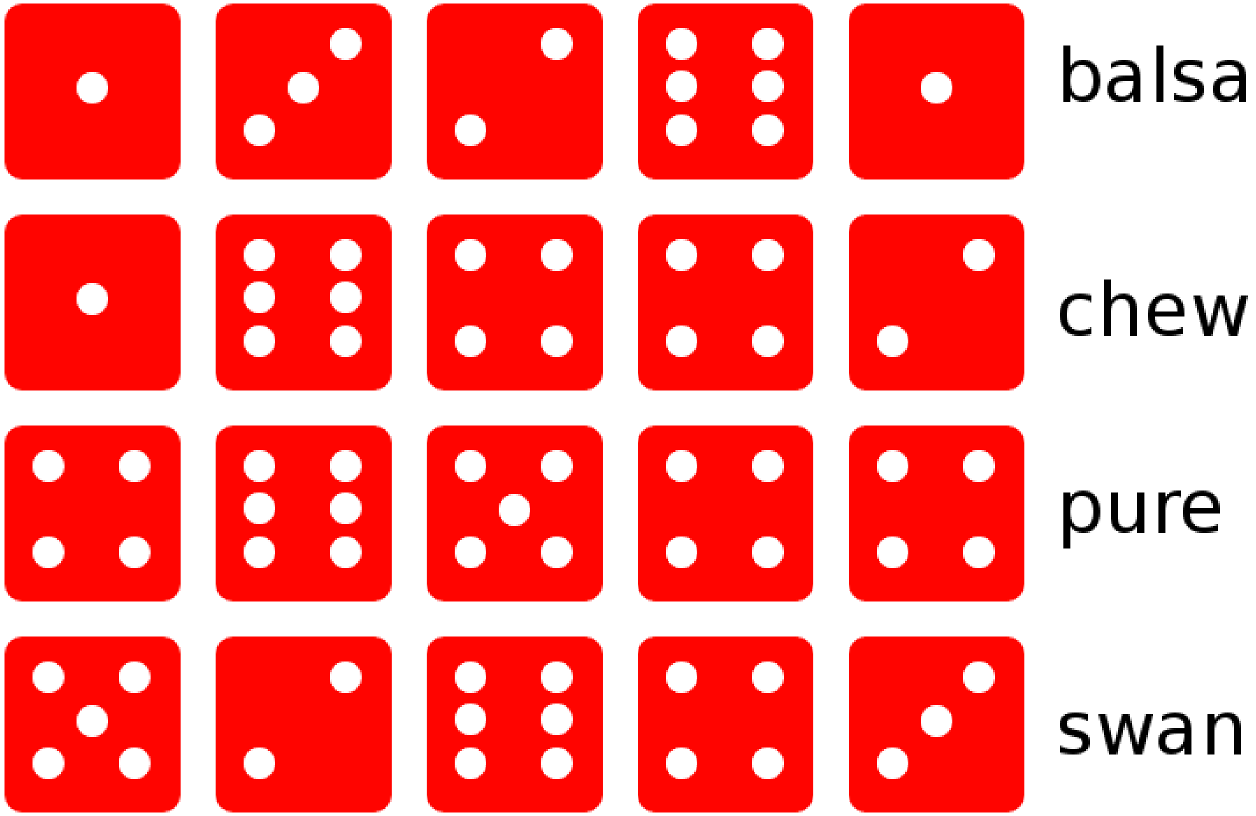
Radically Open Security



Contents

1. Introduction
2. Prior Work
3. Research Question
4. Methods
5. Results
6. Conclusions
7. Future work

What are passphrases?



- balsa chew pure swan
- I have got a nice bike

Prior Work

- 2012 Labrande
  Hybrid dictionary attack

- 2016 Sparell and Simovits
  Markov chains

- 2017 Gaastra, Gijtenbeek and Gommans
  Using lyrics and famous quotes

Research Question

How can software efficiently generate likely passphrases, to be used in passphrase cracking?

- Efficient:
  - Computational power
  - RAM
  - Storage
- Likely:
  - Results

Contribution

- Implement a new method

- Compare different methods
  - Make previous work directly comparable

Hybrid dictionary attack

- Reproduced Labrande‘s work
  - Training dataset
  - Effectiveness comparison

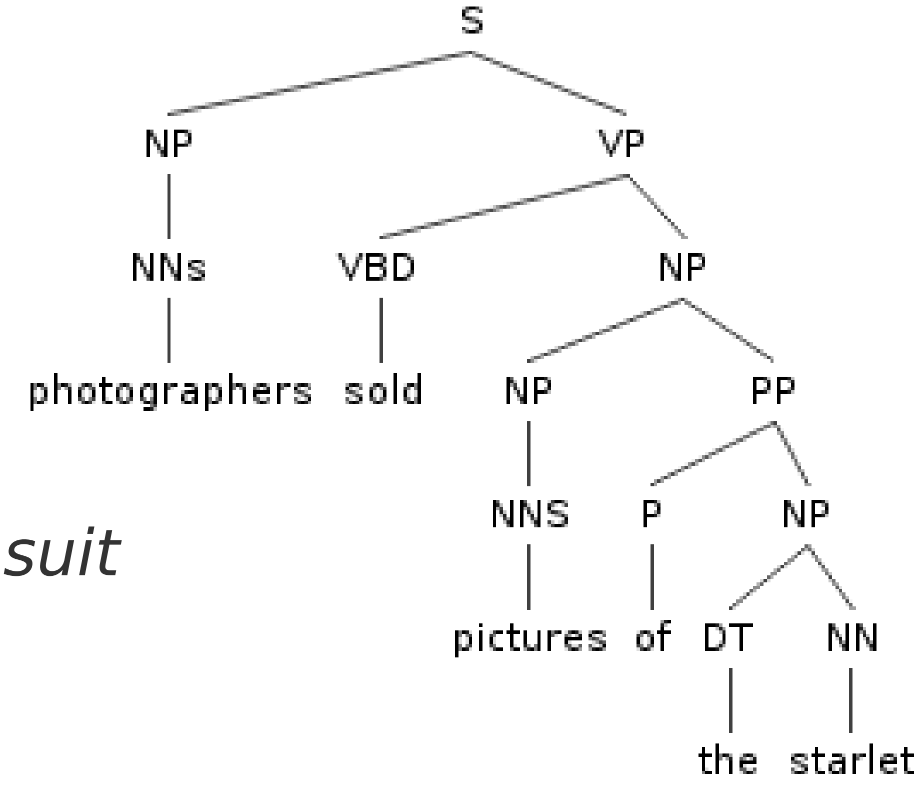
- Dictionary of phrases + sets of rules
  - Lowercase all, remove spaces, etc.

Probabilistic Method Selection

- Markov chains
  done by Sparell and Simovits
- Probabilistic Context-Free Grammar
  applied to passwords successfully
- N-grams
  popular in text prediction

Context-Free Grammar

S → NP VP
NP → Det N | W
VP → V NP
W → I | he | she | Joe
Det → a | the | my | his
N → elephant | cat | jeans | suit
V → kicked | followed | shot



- I followed Joe
- *a* cat shot my elephant

Probabilistic Context-Free Grammar

- *NP → 0.7(Det N) | 0.3(W)*

- Generate probabilities and rules based on texts
  - Word classification database

N-grams

- „have we lost or have we won“ n=2
  - 2 have we
  - 1 we lost
  - 1 lost or
  - 1 or have
  - 1 we won

→ have we won

N-grams

- Generated weighted statistics from:
  - Wikipedia articles
  - Previously cracked passphrases
- Cracking by taking the most frequently occurring n-gram and finding continuations

Results
Effectiveness

- Hybrid dictionary (Labrande)
  - **4.2M** phrases of Korelogic (**200k** of ≥16 characters)
- Hybrid dictionary (ours)
  - **2.3M** phrases of Korelogic (**147k** of ≥16 characters)
  - **1.3M** phrases of LinkedIn (**13k** of ≥16 characters)
- Markov chains
  - **25k** phrases of LinkedIn (**384** of ≥16 characters)
- N-grams
  - **835k** phrases of Korelogic (**33k** of ≥16 characters)
  - **482k** phrases of LinkedIn (**4k** of ≥16 characters)

Results
Efficiency

- Hybrid dictionary
  - Speed: >10 000 000 pps (phrases per second)
  - Storage: medium (690MiB)
- Markov chains
  - Speed: 2 500–22 500 pps
  - Storage: unknown
- N-grams
  - Speed: 3 300 000 pps
  - Storage: low-medium (47-464MiB)

Conclusions

- Hybrid dictionary is efficient and effective
- N-grams most effective when length of phrase $\leq n$

Future work

- Better language modeling using n-grams
- Probabilistic Context-Free Grammar
- Neural Networks

Thank you

- Thanks to Radically Open Security

- See our git repository for GPLv3 licensed:
  - N-gram phrase generator & models (n=2 and n=3)
  - Phrase dictionary & rules
  - Slides and preview of the paper

  github.com/radicallyopensecurity/passphrase-cracking

- Questions?