NTNU

Norwegian University of Science and Technology

Constrained approximate search in misuse-based intrusion detection

Ambika Shrestha Chitrakar
Supervisor: Prof. Slobodan Petrovic

FINSE
10th of may 2017

Contents

Introduction

- Snort: a misuse-based intrusion detection
  - Detects intrusions based on attack signatures stored as rules
  - One of the ways to detect attacks is by matching the payload information of the network traffic with the content field of the Snort rules
  - Uses Aho-corasick (exact search)

- Problem with Snort:
  - Snort fails to detect new attacks
  - Moreover, same attacks with small changes in the attack pattern can also evade Snort

- Proposed solutions:
  - Approximate search?
  - What about constrained approximate search?

Background

- Approximate search:
  - Allows some level of errors/tolerance to find the occurrences of the search pattern in the given string
  - Uses distance functions such as hamming distance, Lavenshtein distance
  - Given string T=abbaccacbbadrbbb, and pettern P = bbba, find all the occurrences of P in T with errors k=1, using edit distance
    - abbaccacbbadrbbb - occurrences at position 4, 11, and 16
  - Application: digital forensics, text-retrieval, computational biology etc.

Background

- Constrained approximate search:
  - More precise than approximate search
  - Errors can be defined on the type of edit operation
    - Only substitutions, only deletions and substitutions, only insertions and substitutions etc
  - Errors can also be defined on the allowed number of each edit operations
    - If k=5, insertions=1, deletions=2, substitutions=2

- When to use constrained approximate search?
  - When one knows the probability of errors and want to be more precise than unconstrained approximate search
  - Given a set of strings T: {threat, thrett, treat} and pattern P: threat, find all the occurrences of P in T, with errors k=1 and constraint only 1 substitution
    - Matches threat with 0 error
    - Matches thrett with one character substitution
    - No match with treat, but its a match when unconstrained approximate search is applied

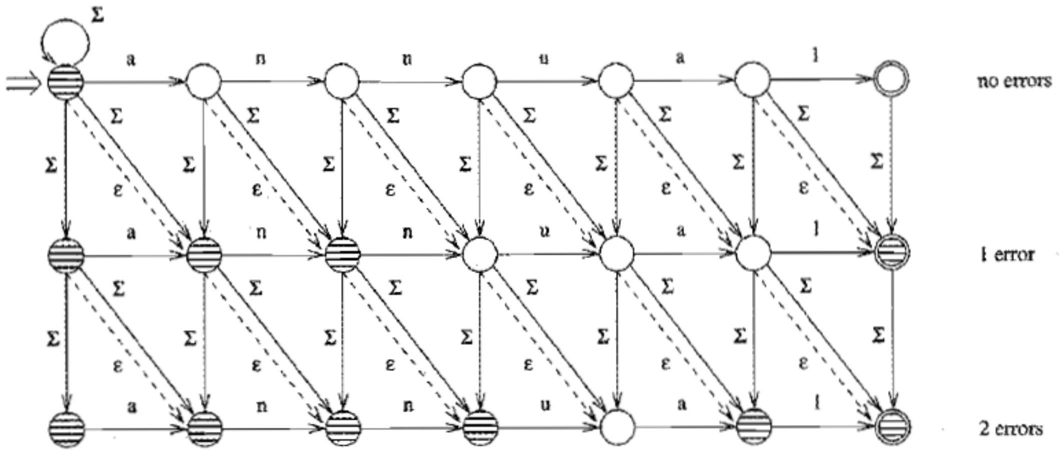
NTNU

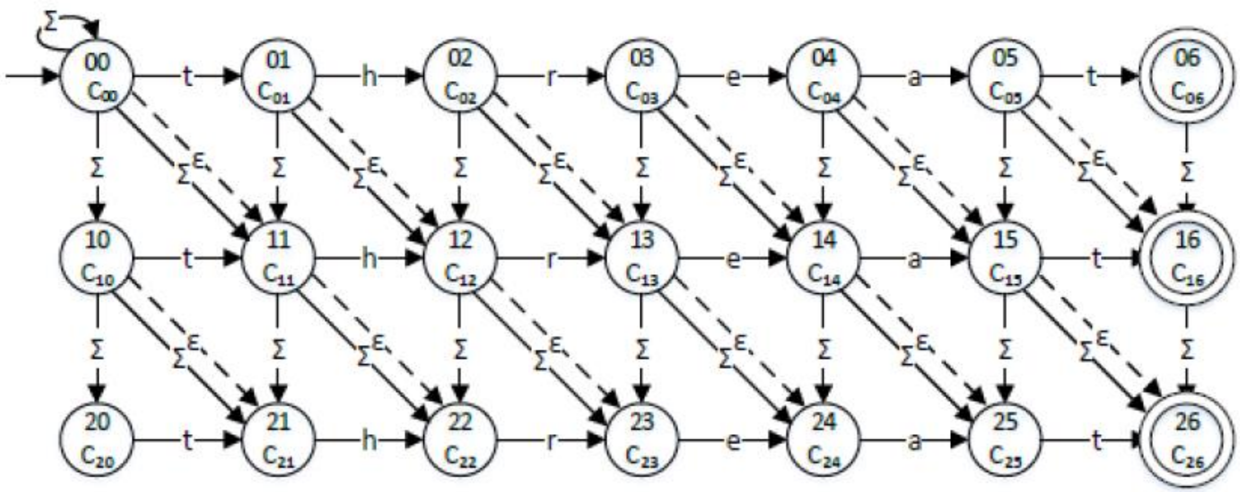
Related work

- Constraints on indels: Sankoff-Indels
  - Based on dynamic programming
- Constraints on indels: CRBP-Indels
  - based on automata theory
- Constraints on each edit operations: CRBP-OpCount
  - Based on automata theory

CRBP-OpType and CRBP-OpCount

- Based on Row-wise Bit-Parallel algorithm by Wu and Manber

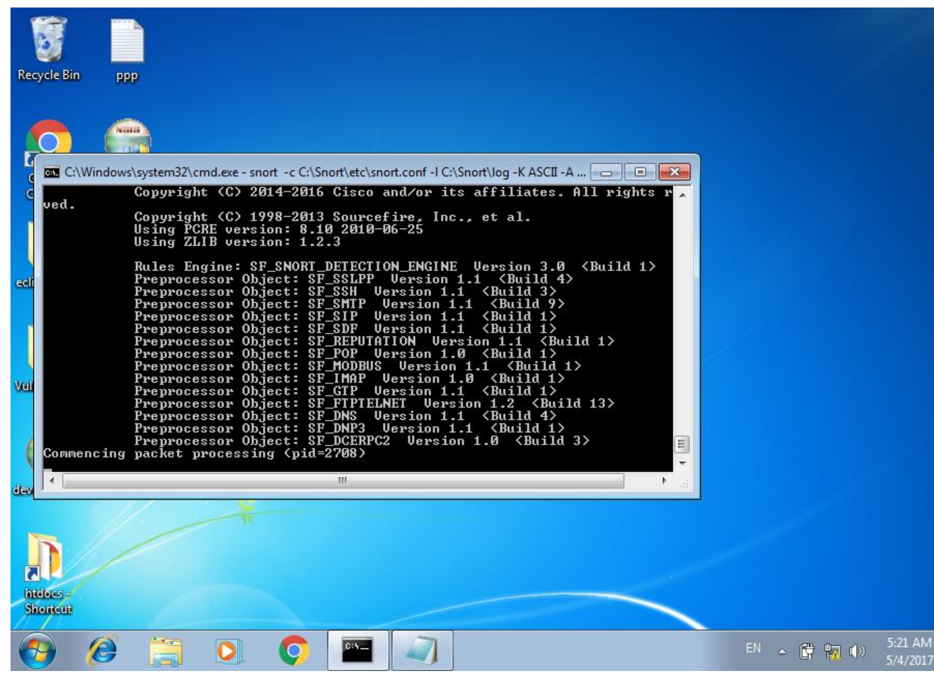




NTNU

Experiment

Attacker machine

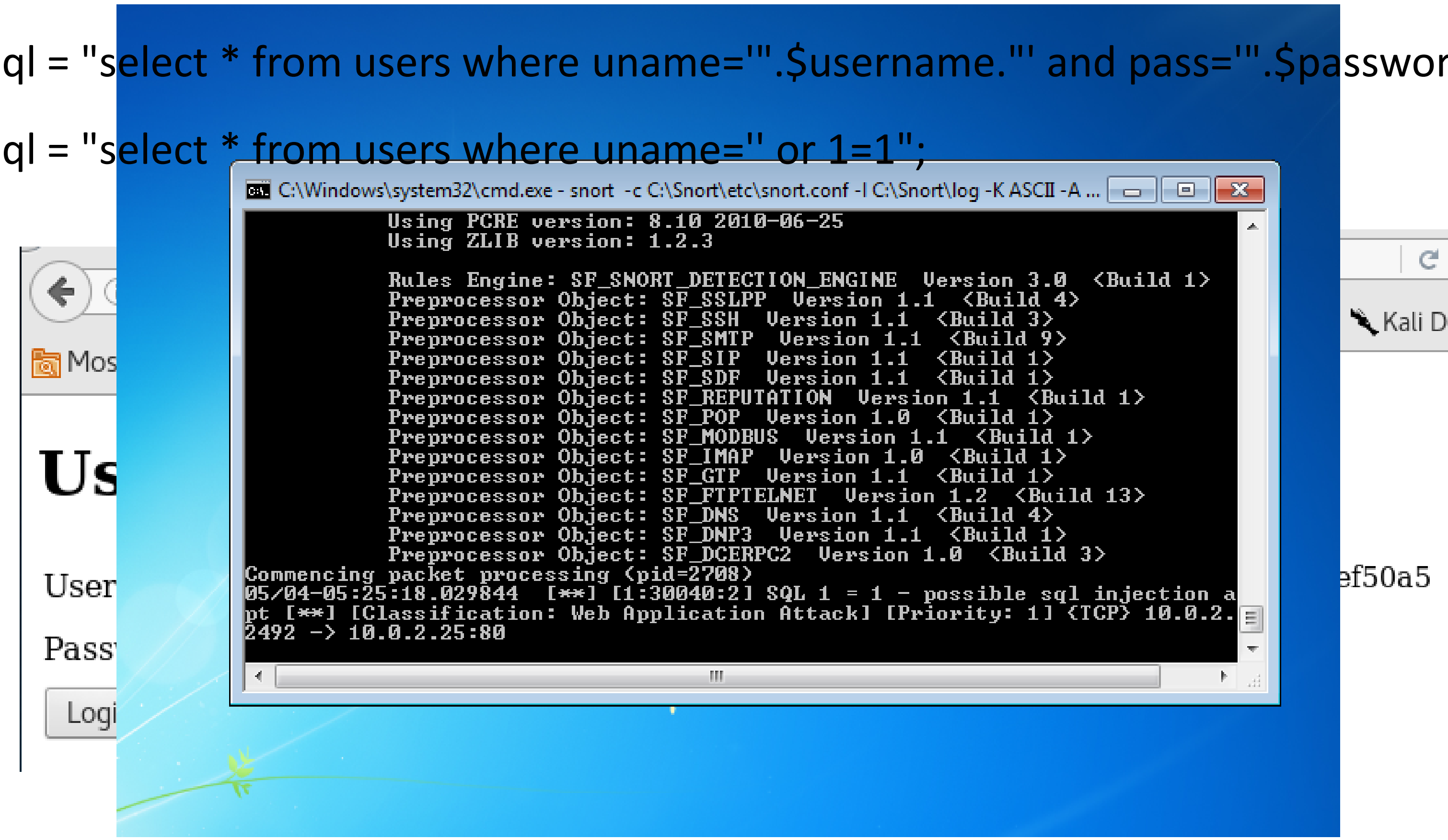


Victim machine (web server)

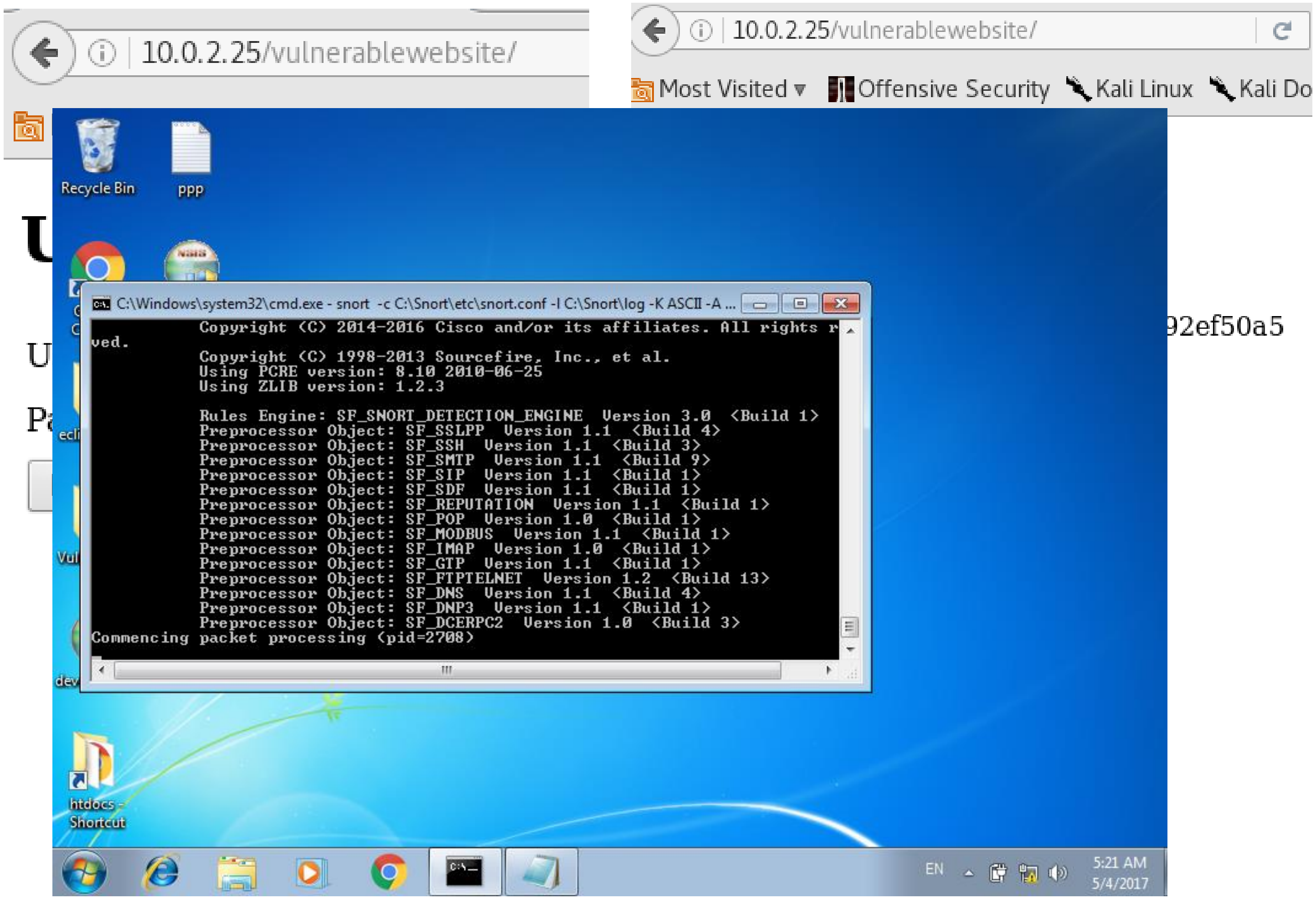
Experiment

$sql = "select * from users where uname='".$username."' and pass='".$password."'";

$sql = "select * from users where uname='' or 1=1";



Experiment


10.0.2.25/vulnerablewebsite/
10.0.2.25/vulnerablewebsite/
Most Visited
Offensive Security
Kali Linux
Kali Do
92ef50a5
Recycle Bin
ppp
htdocs - Shortcut
C:\Windows\system32\cmd.exe - snort -c C:\Snort\etc\snort.conf -l C:\Snort\log -K ASCII -A ...
Copyright (C) 2014-2016 Cisco and/or its affiliates. All rights r
ved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.3
Rules Engine: SF_SNORT_DETECTION_ENGINE Version 3.0 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Commencing packet processing (pid=2708)
EN
5:21 AM
5/4/2017

Results

	Pattern	k	Total	TP	FP	TN	Time
	1=1	2	200	10	190	0	10
RBP	'1'='1'	4	1813	182	1619	12	43

	Pattern	k	Total	TP	FP	TN	Time
	1=1	s=2	200	10	159	31	9
CRBP-OpType	'1'='1'	is=4	1813	182	1594	37	33

	Pattern	k	Total	TP	FP	TN	Time
	1=1	s=2	200	10	159	31	21
CRBP-OpCount	'1'='1'	i=2,s=2	1813	182	977	654	144

NTNU

Discussion

- Constrained and unconstrained search algorithms can be used to detect new similar attacks
- Unconstrained approximate search can generate lot of false positives
- CRBP-OpType and CRBP-OpCount algorithms can be used to reduce the number of false positives
- Better to use CRBP-OpType algorithm if attacks can be detected by specifying the type of edit operations
- Better to use CRBP-OpCount if we know the probability of changes in each edit operations
- CRBP-OpCount is complex compared to CRBP-OpType, due to use of counters in each states

NTNU

Conclusion

- Exact search is important when attack signatures does not vary for a particular attack
- Unconstrained approximate search is useful when attack signature can vary by some edit operations and probability of error type is unknown
- The constrained approximate search can be used when probability of error types is known

Thank you!