Fuzzing E-mail Filters with Generative Grammars and N-Gram Analysis

Presented by

Sean Palka / George Mason University

And

Damon McCoy / International Computer Science Institute

WOOT 2015

/bin/whoami

- Graduate Student at George Mason University
- Senior Penetration Tester at Booz Allen Hamilton
- Social Engineering Researcher

Acknowledgements…

This research could not have been accomplished without the assistance of:

- Dr. Damon McCoy
- Dr. Harry Wechsler
- Dr. Mihai Boicu
- Dr. Dana Richards
- Dr. Duminda Wijesekera
- George Mason Department of Computer Science
- Booz Allen Hamilton

Current Phishing Landscape

- Phishing is no longer just a broad spectrum attack.
- Highly evolved, targeted attack strategies
  - Phishing, Smishing, Twishing, Whaling, Spear-phishing….
- Open-source attack frameworks
  - Social engineering toolkit (SET), Phishing Frenzy, Wifiphisher…
- Threat has evolved, but so has detection

Phishing Detection and Prevention

User-Centric Models

- Detected attacks and crafted examples used in awareness training
- Modified examples used as payloads in live exercises and simulations

Technical Models

- Known examples used as training datasets
- Identification of threat signatures using various analysis techniques

Typical Email Filtering

Keyword Filtering

- Triggers on specific phrases or keywords regardless of context
- Signature-based approach, not very flexible
- Suffers from same limitation as black-listing in other media

Bayesian Models

- Determines threat based on word probabilities
- Each word contributes to the overall threat score
- Requires training on known good and bad e-mails to be effective

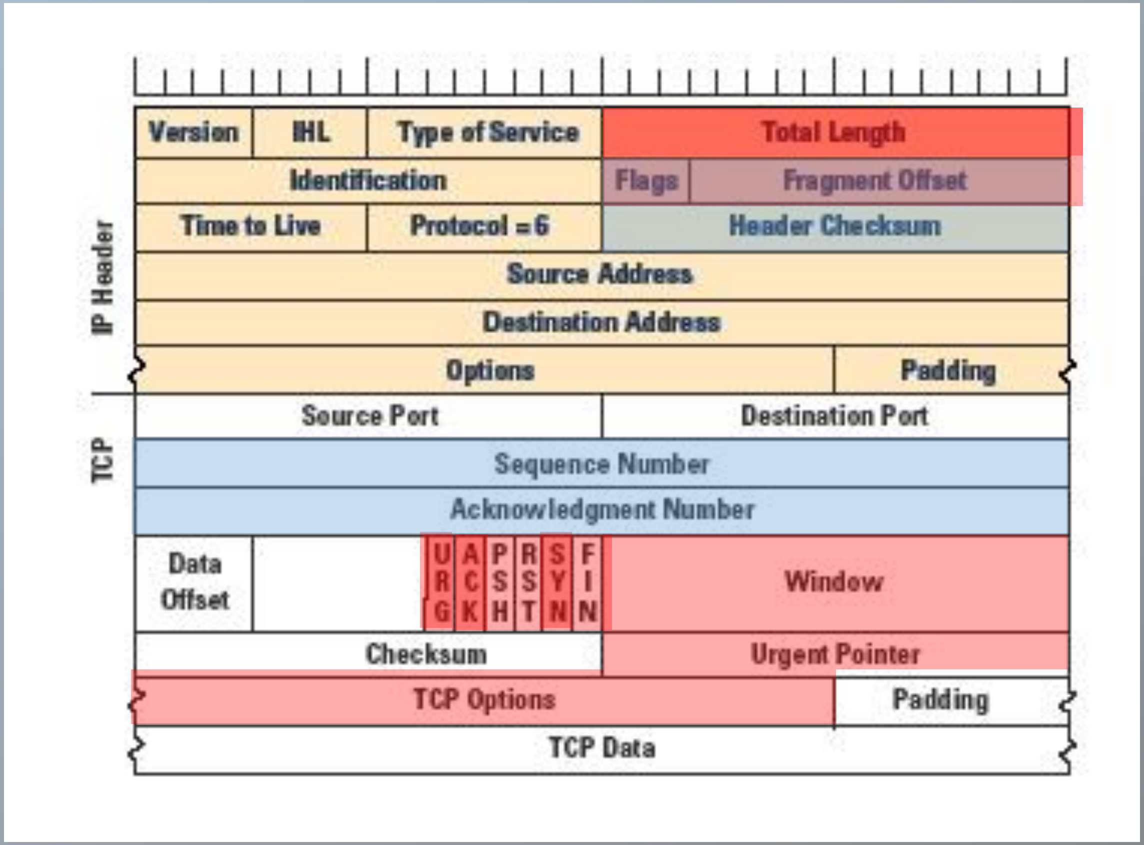
Goal

- Defensive: Given the number of potential e-mail variations, how can we evaluate whether a given filtering approach is effective?
- Offensive: Can we figure out a way to increase the odds of an attack succeeding by finding kinks in the armor?

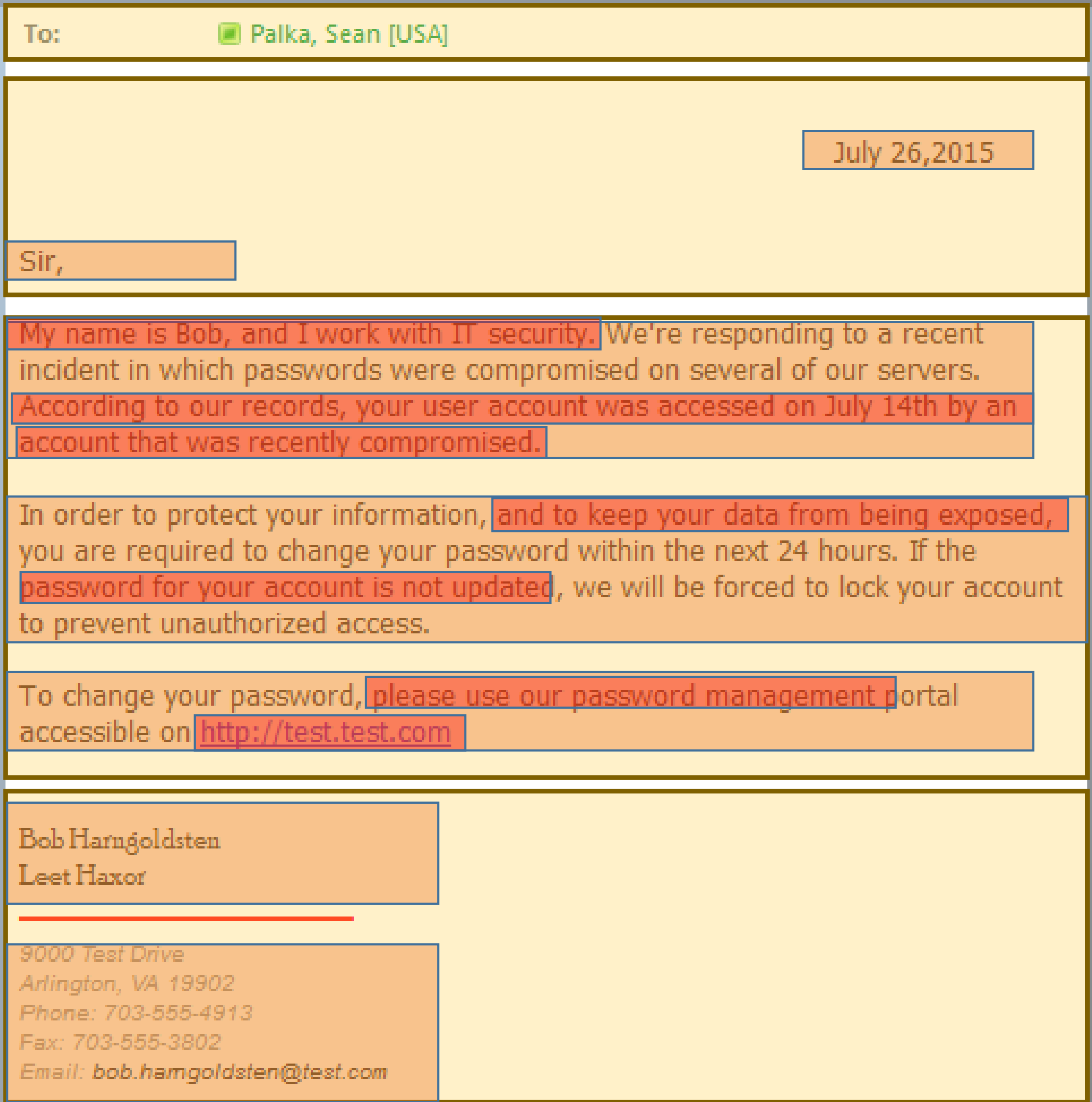
- **Answer: Fuzzing**

Fuzzing Overview

- Vary input to identify boundary conditions that may be exploitable
- Basic Example: TCP/IP packet fuzzing



E-mail Variation



Headers

To: Palka, Sean [USA]

Start

Date

July 26,2015

Salutation

Sir,

Middle

Introduction

My name is Bob, and I work with IT security. We're responding to a recent incident in which passwords were compromised on several of our servers. According to our records, your user account was accessed on July 14th by an account that was recently compromised.

Threat

In order to protect your information, and to keep your data from being exposed, you are required to change your password within the next 24 hours. If the password for your account is not updated, we will be forced to lock your account to prevent unauthorized access.

Action

To change your password, please use our password management portal accessible on http://test.test.com

End

Name

Bob Harngoldsten
Leet Haxor

Address

9000 Test Drive
Arlington, VA 19902
Phone: 703-555-4913
Fax: 703-555-3802
Email: bob.harngoldsten@test.com

Building an e-mail

- Previously we used generative grammars to dynamically create useful phishing e-mail contents for exercises (PhishGen)

- By varying the different production rules, we cause variations in the different sections and subsections in the e-mail

- Our original approach was used to avoid repetition in e-mails for exercises, and the same approach works for intelligent fuzzing

Example of Production Rules and Placeholders

ID	Left Rule	Right Rule
1	{START}	{INTRO}{PROBLEM}{RESOLVE}
2	{INTRO}	{Hello, [FIRSTNAME].}
3	{PROBLEM}	{Your hasEmployee() is invalid.}
4	{PROBLEM}	{Your hasEmployee() has a hasMisc(hasEmployee([X])).}
5	{RESOLVE}	{Please click here to have your hasEmployee([X]) updated.}
6	{RESOLVE}	{Please check your hasEmployee([Y]) to ensure there are no issues.}

Expansion Example

{START}

⇩ ***Expand {START} using production rule 1***

{INTRO}{PROBLEM}{RESOLVE}

⇩ ***Expand {INTRO} using production rule 2***

{Hello, [FIRSTNAME].}**{PROBLEM}{RESOLVE}**

⇩ ***Expand {PROBLEM} using production rule 4***

{Hello, [FIRSTNAME].} {Your hasEmployee() has a hasMisc(hasEmployee([X])).}
{RESOLVE}

⇩ ***Expand {RESOLVE} using production rule 5***

{Hello, [FIRSTNAME].} {Your hasEmployee() has a hasMisc(hasEmployee([X])).} {Please click here to have your hasEmployee([X]) updated.}

⇩ ***Remove {} delimiters***
Apply relevant values to global and relational placeholder variables

Hello, Bob. Your computer has a virus. Please click here to have your computer updated.

Signatures

- Each generated e-mail has a “signature” defined by the production rules that were used to create it.

- Previous example:

  1→2 → 4 → 5 → G1 → R1 → R2

- Previous grammar could also have generated:

  1→2 → 3 → 6 → G1 → R2

  1→2 → 3 → 6 → G1 → R1

Identifying Filtered Rules

- If we sent the previous e-mail, and it was filtered, how could we determine which rule (or combination or rules) resulted in the filtering?

- What if a different variations was not filtered?

FILTERED: 1 → 2 → 4 → 5 → G1 → R1 → R2

UNFILTERED: 1 → 2 → 3 → 6 → G1 → R2

1 → 2 → 3 → 6 → G1 → R1

N-Grams

$$\begin{tikzcd}\boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{5} \arrow[r] & \boxed{G1} \arrow[r] & \boxed{R1} \arrow[r] & \boxed{R2}\end{tikzcd}$$

<u>N=1</u>

1

2

4

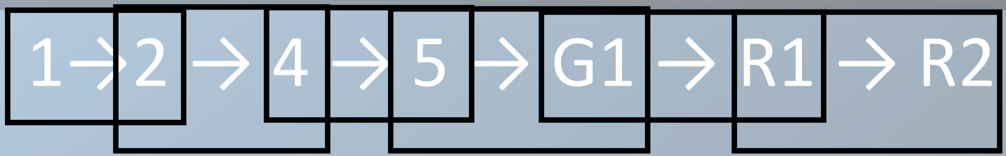
5

G1

R1

R2

N-Grams



<u>N=1</u>	<u>N=2</u>
1	1→2
2	2→ 4
4	4→ 5
5	5 → G1
G1	G1 → R1
R1	R1 → R2
R2	

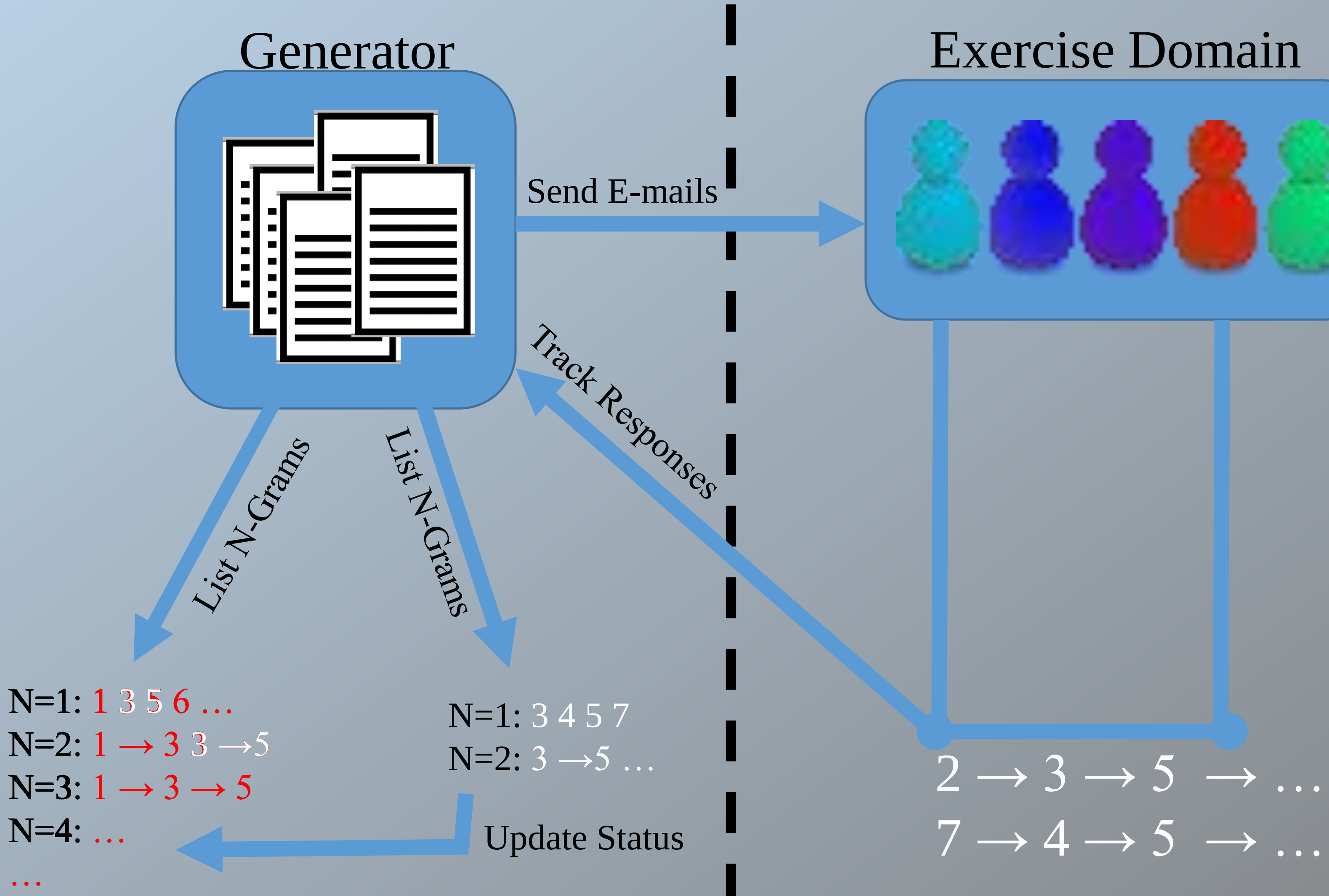
N-Grams

1→2 → 4 → 5 → G1 → R1 → R2

N=1	N=2	N=3
1	1→2	1→2 →4
2	2→ 4	2→ 4 →5
4	4→ 5	4→ 5 →G1
5	5 → G1	5 → G1 →R1
G1	G1 → R1	G1 → R1 →R2
R1	R1 → R2	
R2		

N=3 , N=4, N=5 …..

Fuzzing Strategy



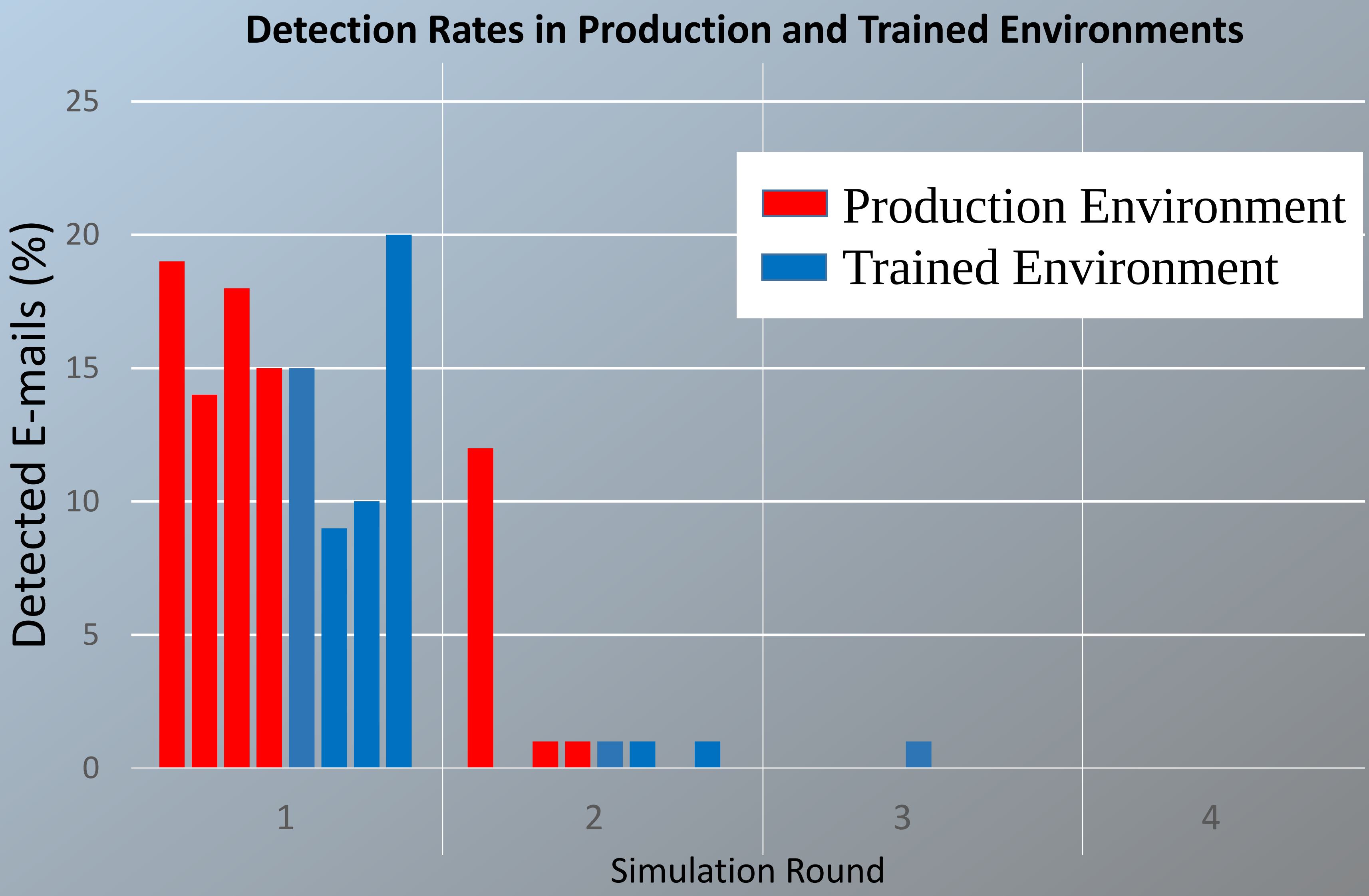
Known-good production rules are favored in future generations

Simulations

- To test our approach, we ran simulations in two different environments:
  - Production environment supporting several thousand users with existing detection measures
  - Trained environment using SpamAssassin and Bayesian probabilistic classification (795,092 training samples)

- For each environment, we ran 4 rounds of simulations. Each had 4 sets of 100 generated e-mails, and used feedback from the exercise domain to update production rules

Results


Detection Rates in Production and Trained Environments
Detected E-mails (%)
25
20
15
10
5
0
1
2
3
4
Simulation Round
🟥 Production Environment
🟦 Trained Environment

Conclusions

- After 4 rounds of testing, our generator was able to bypass all detection filters and get all 100 e-mails through to the inbox
- Successful but very noisy approach, better suited for administrators than attackers
- To request a copy of PhishGen, please send an e-mail to spalka (at) gmu.edu with subject line: Phishgen Request

Questions