Leveraging Cross-Website Coordination to Mitigate Credential Stuffing

Michael K. Reiter

Lawrence M. Slifkin Distinguished Professor
University of North Carolina at Chapel Hill

*Joint work with **Ke Coby Wang**.*

1

Password Reuse :: Definition

same user,
same or similar password,
multiple websites.

2

2

Leaked Passwords

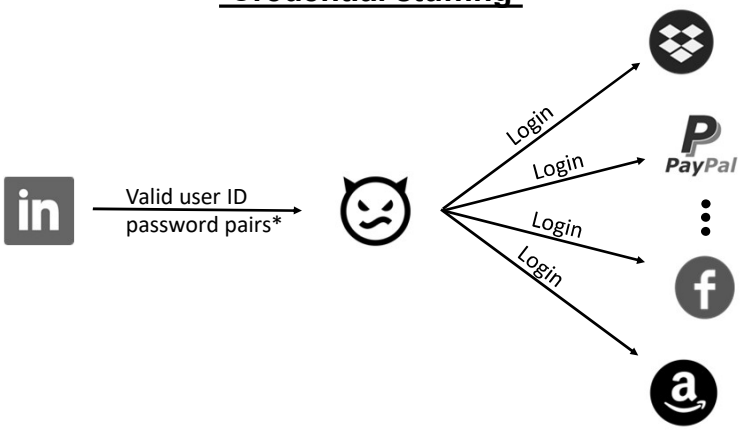
Announced data breaches in the **past year**:

3

3

Password Reuse :: Threats

"Credential stuffing"



* via database breaches, phishing, malware, social engineering, etc.

4

4

The reuse of passwords is the No. 1 cause of harm on the internet.

--- Alex Stamos (former CSO, Facebook)

99% of compromised user accounts come from password reuse.

--- Patrick Heim (Head of Trust & Security, Dropbox)

Credential stuffing is enormously effective due to the password reuse problem.

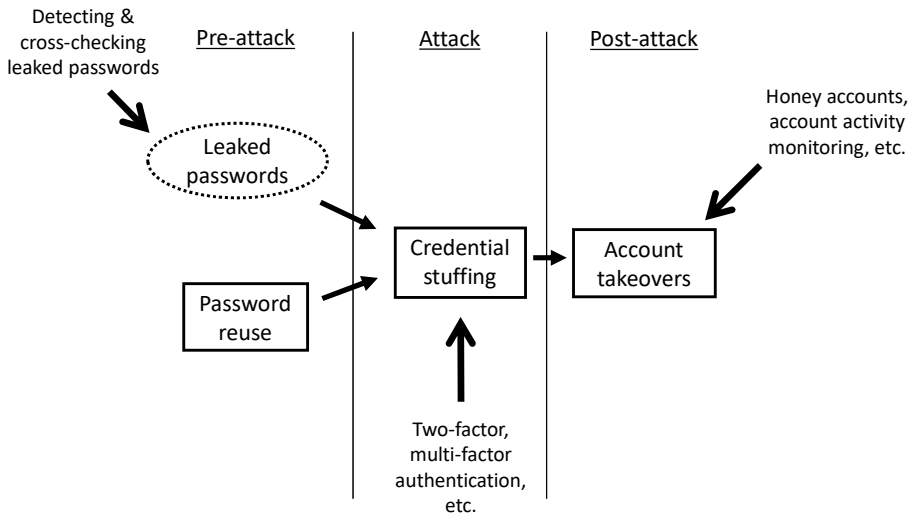
--- Troy Hunt (Regional Director, Microsoft)

* via database breaches, phishing, malware, social engineering, etc.

5

5

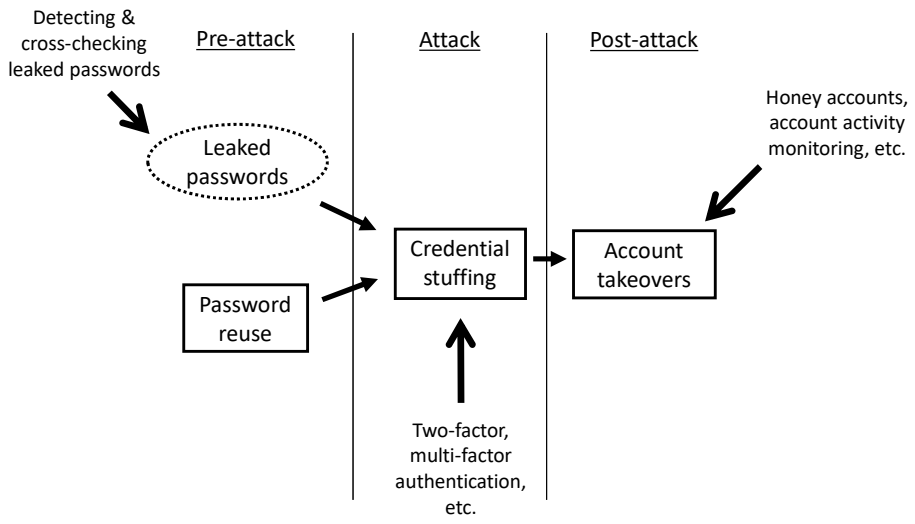
Existing Works



6

6

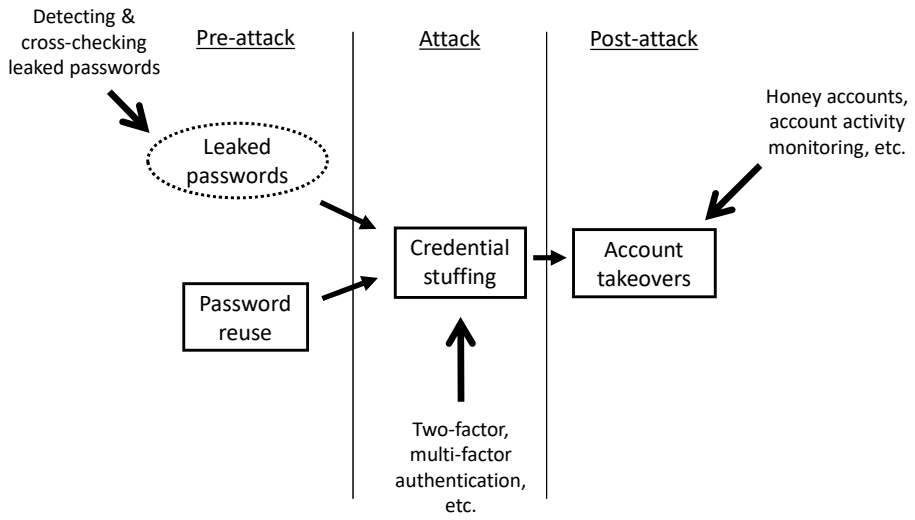
Existing Works



7

7

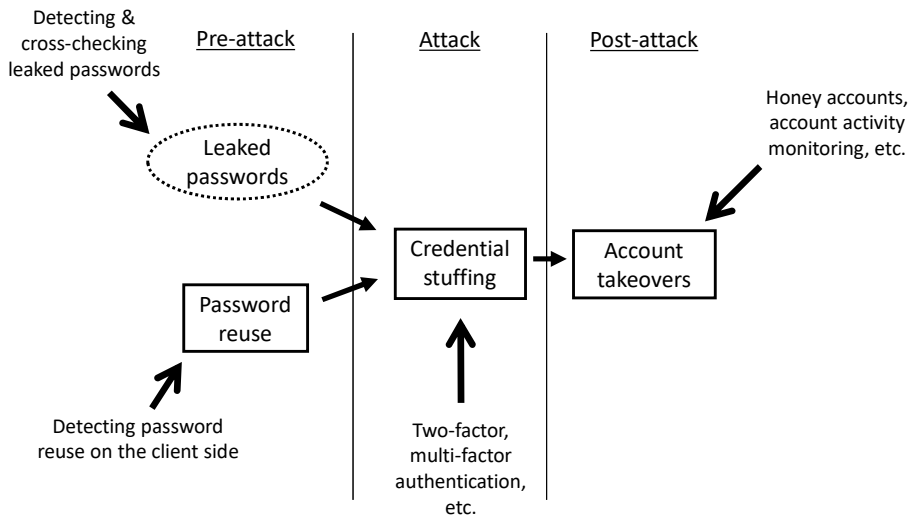
Existing Works



8

8

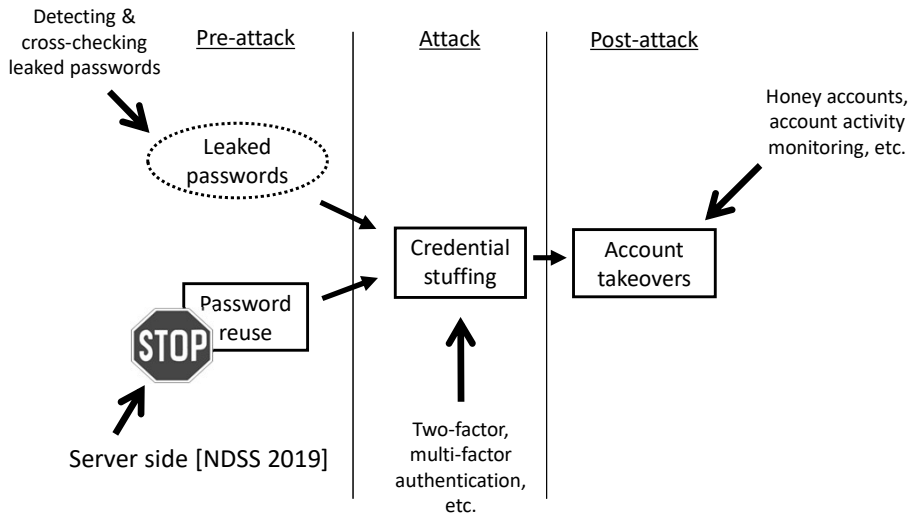
Existing Works



9

9

Our Work



10

10

Interfering with Password Reuse ::

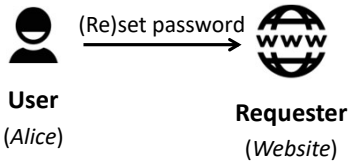
Goals :: Functionality

11

11

Interfering with Password Reuse ::

Goals :: Functionality

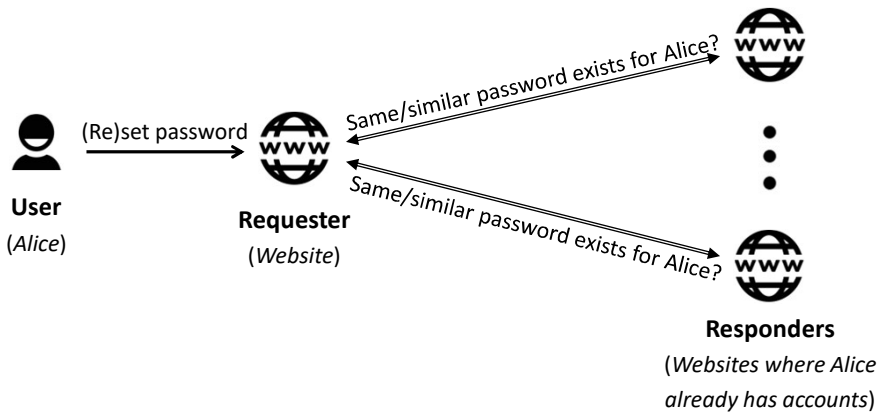


12

12

Interfering with Password Reuse ::

Goals :: Functionality

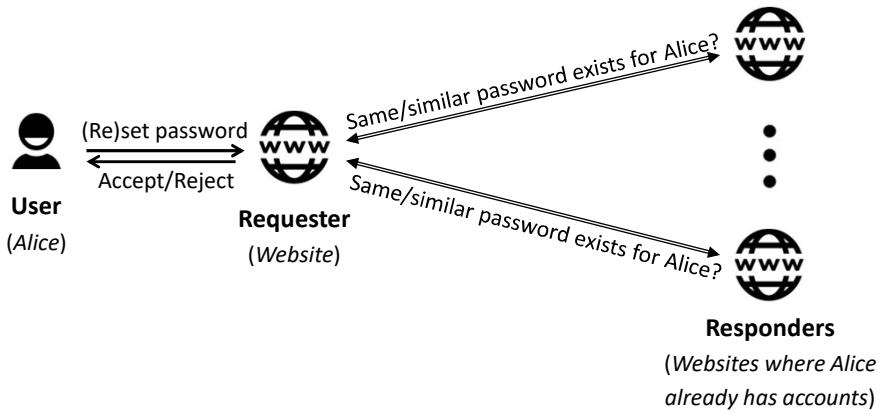


13

13

Interfering with Password Reuse ::

Goals :: Functionality

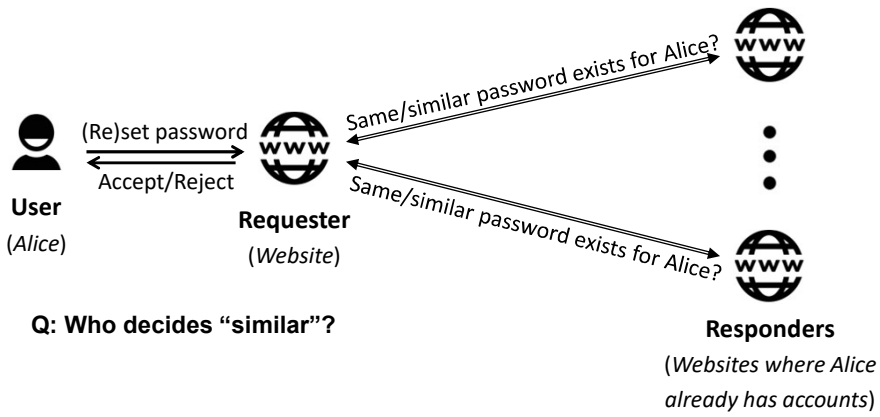


14

14

Interfering with Password Reuse ::

Goals :: Functionality



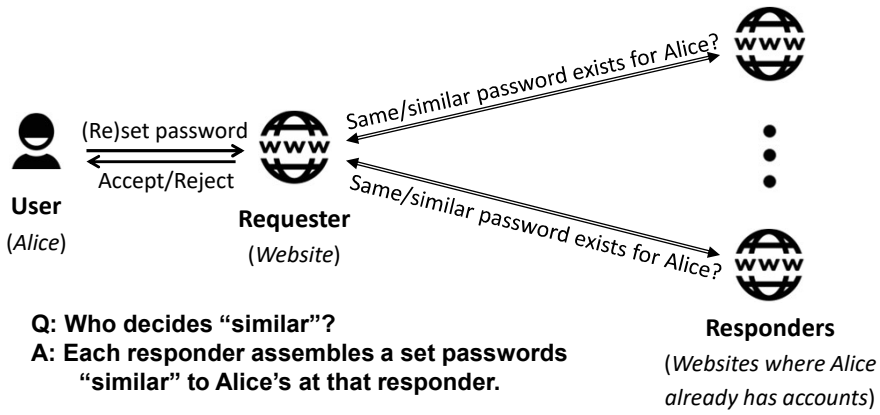
Q: Who decides "similar"?

15

15

Interfering with Password Reuse ::

Goals :: Functionality



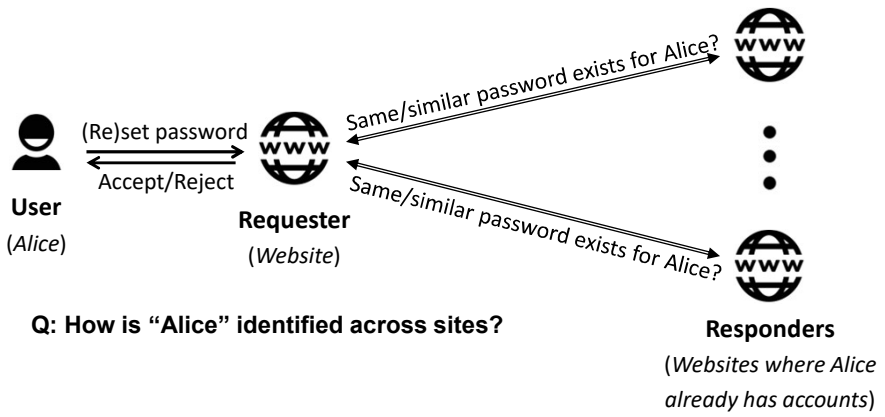
Q: Who decides "similar"?
A: Each responder assembles a set passwords "similar" to Alice's at that responder.

16

16

Interfering with Password Reuse ::

Goals :: Functionality



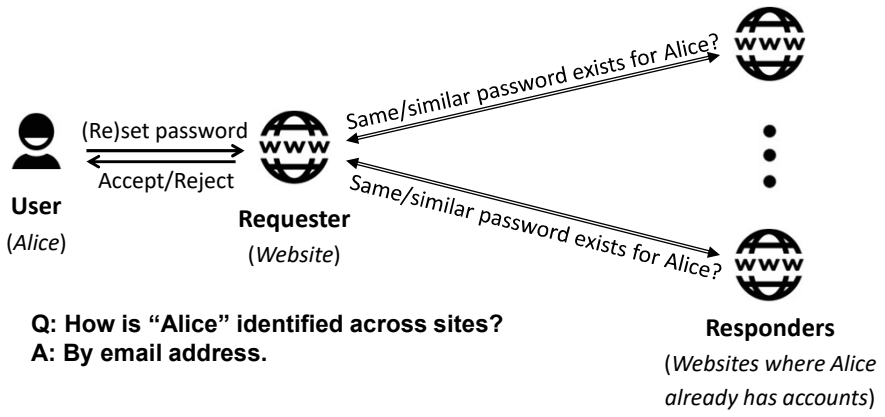
Q: How is "Alice" identified across sites?

17

17

Interfering with Password Reuse ::

Goals :: Functionality



Q: How is "Alice" identified across sites?
A: By email address.

18

18

Interfering with Password Reuse ::

Goals :: Deployment

- We don't require a universal adoption of our framework
- A simple estimate suggests that if these 20 websites adopted our framework, then each Internet user would have ~4-5 different and dissimilar passwords

"If multiple passwords cannot be avoided, four or five is the maximum for unrelated, regularly used passwords that users can be expected to cope with" [1]

Website	Users (M)	Websites	Users (M)
Facebook	2167	Taobao	580
YouTube	1500	Outlook	400
WhatsApp	1300	Sina Weibo	376
Yahoo!	1000	Twitter	330
Gmail	1000	Amazon	310
WeChat	980	Baidu Tieba	300
QQ	843	LinkedIn	260
Instagram	800	Snapchat	255
Tumblr	794	Reddit	250
iCloud	782	Pinterest	200

Table: Top 20 websites ranked by number of active users. In addition, there are **3.58 billion** active Internet users worldwide.

[1] Adams and M. A. Sasse, "Users are not the enemy," Communications of the ACM, vol. 42, pp. 40– 46, Dec. 1999

19

19

Interfering with Password Reuse ::

Goals :: Security and Privacy

20

20

Interfering with Password Reuse ::

Goals :: Security and Privacy

- **Account location privacy**: Participating websites are not disclosed to one another

21

21

Interfering with Password Reuse ::

Goals :: Security and Privacy

- **Account location privacy**: Participating websites are not disclosed to one another
- **Account security**: Interfere with password reuse while not qualitatively degrading account security in other ways

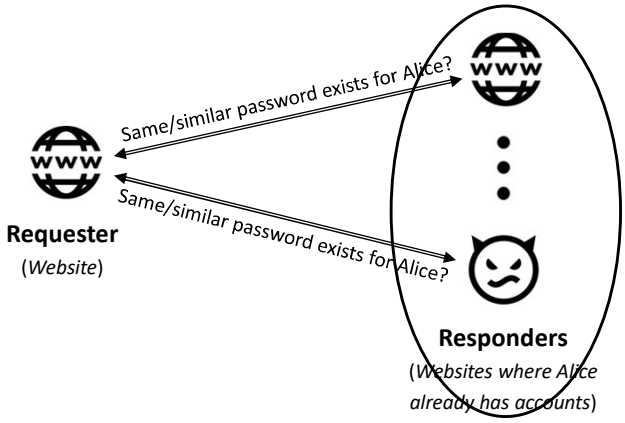
22

22

Interfering with Password Reuse ::

Goals :: Security and Privacy

- **Account location privacy**: Participating websites are not disclosed to one another
- **Account security**: Interfere with password reuse while not qualitatively degrading account security in other ways



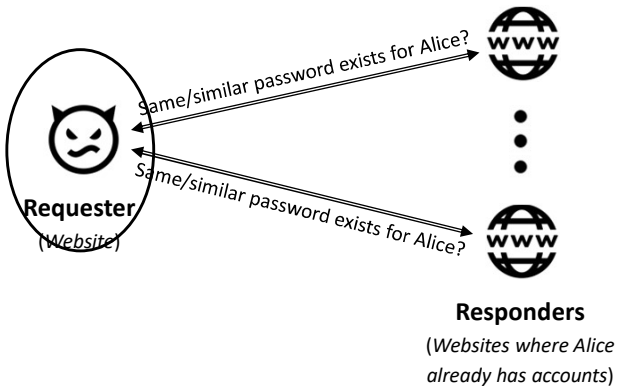
23

23

Interfering with Password Reuse ::

Goals :: Security and Privacy

- **Account location privacy**: Participating websites are not disclosed to one another
- **Account security**: Interfere with password reuse while not qualitatively degrading account security in other ways

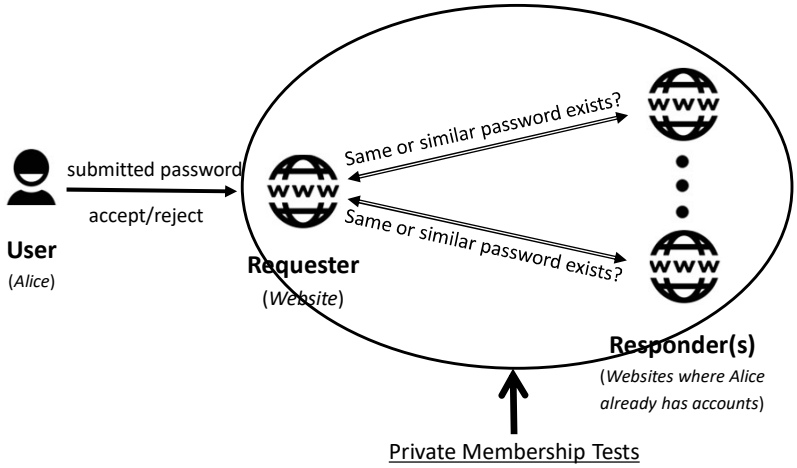


24

24

Interfering with Password Reuse ::

Design :: Private Membership Test



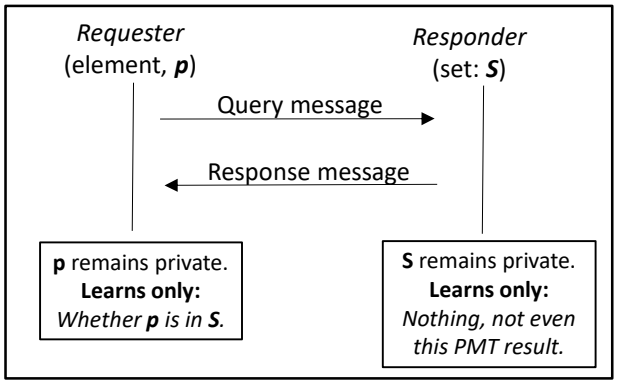
25

25

Interfering with Password Reuse ::

Design :: Private Membership Test

Membership Test: Is *p* in *S* ?

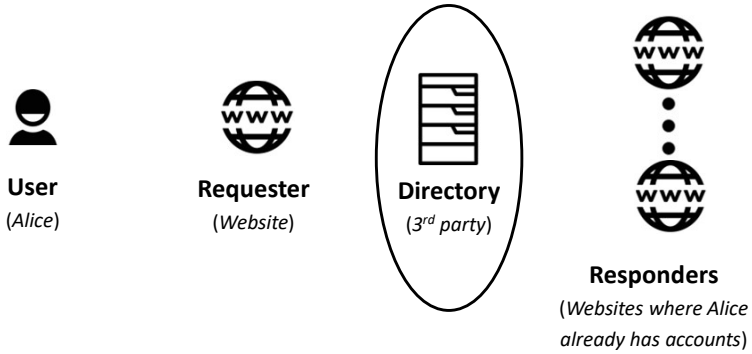


26

26

Interfering with Password Reuse ::

Design :: Directory

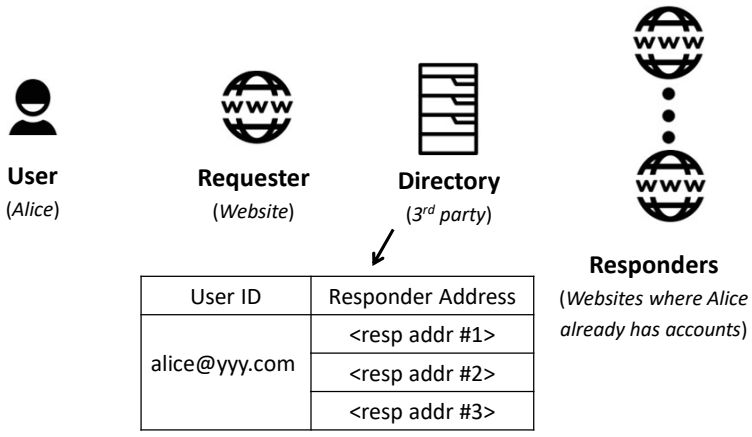


27

27

Interfering with Password Reuse ::

Design :: Directory



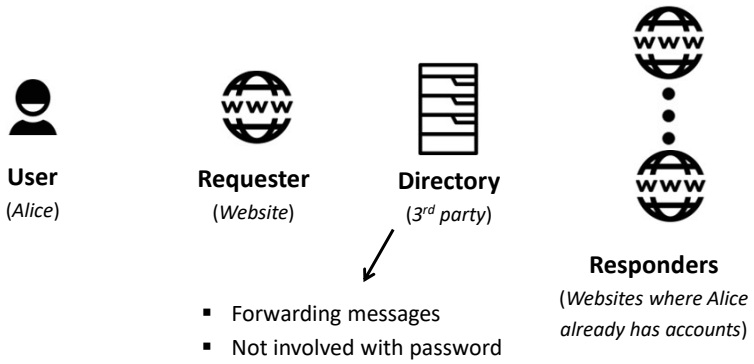
User ID	Responder Address
alice@yyy.com	<resp addr #1>
	<resp addr #2>
	<resp addr #3>

28

28

Interfering with Password Reuse ::

Design :: Directory



- Forwarding messages
- Not involved with password storage or any cryptographic operations for PMTs

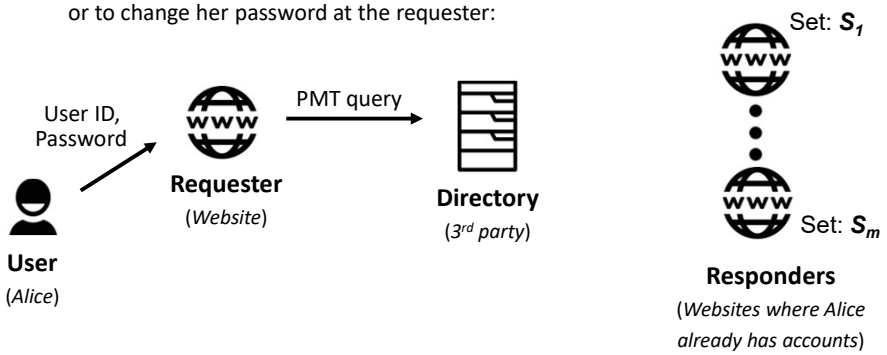
29

29

Interfering with Password Reuse ::

Design :: Directory

When Alice tries to register a new account or to change her password at the requester:



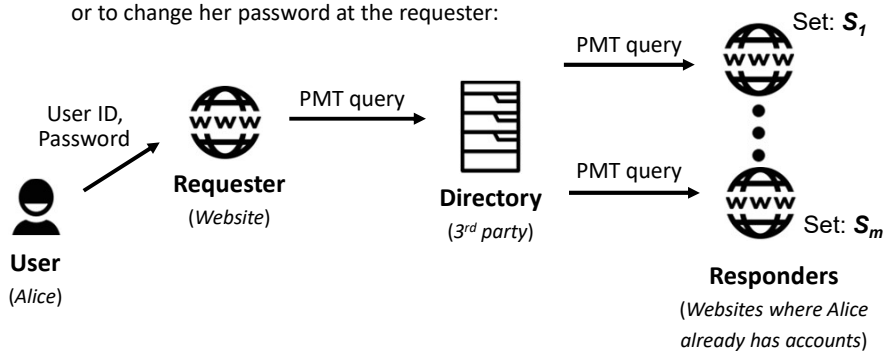
30

30

Interfering with Password Reuse ::

Design :: Directory

When Alice tries to register a new account
or to change her password at the requester:



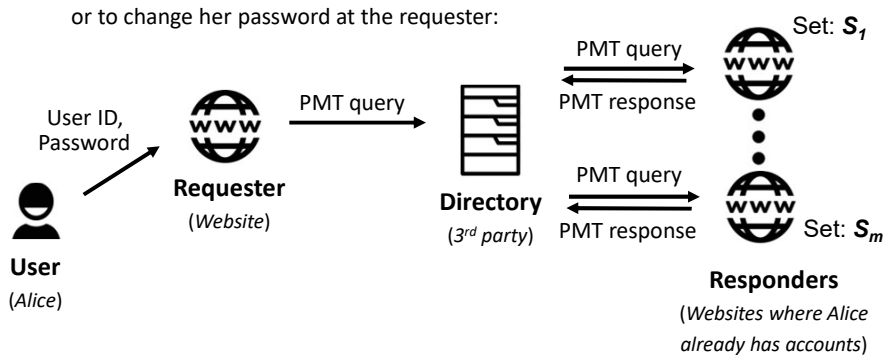
31

31

Interfering with Password Reuse ::

Design :: Directory

When Alice tries to register a new account
or to change her password at the requester:



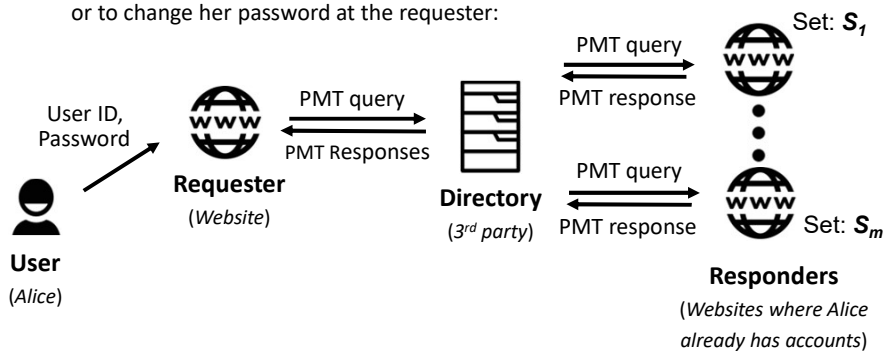
32

32

Interfering with Password Reuse ::

Design :: Directory

When Alice tries to register a new account
or to change her password at the requester:



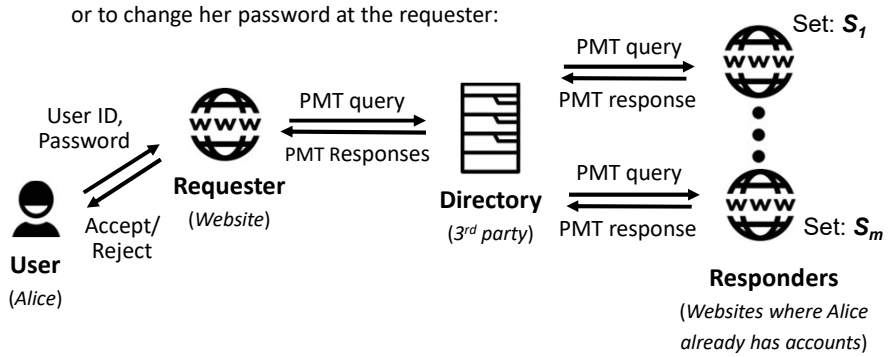
33

33

Interfering with Password Reuse ::

Design :: Directory

When Alice tries to register a new account
or to change her password at the requester:

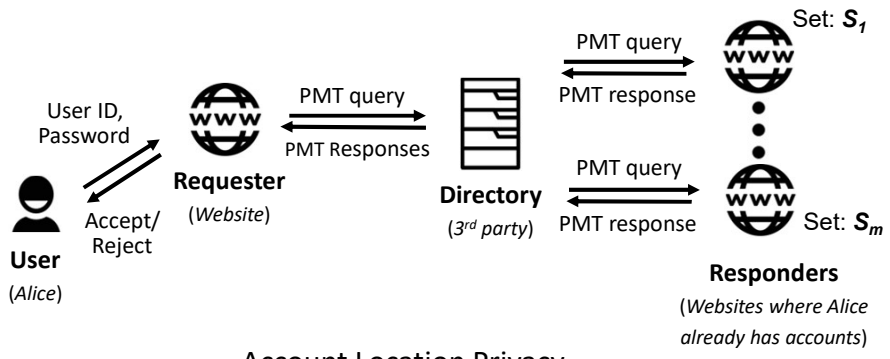


34

34

Interfering with Password Reuse ::

Design :: Account Location Privacy



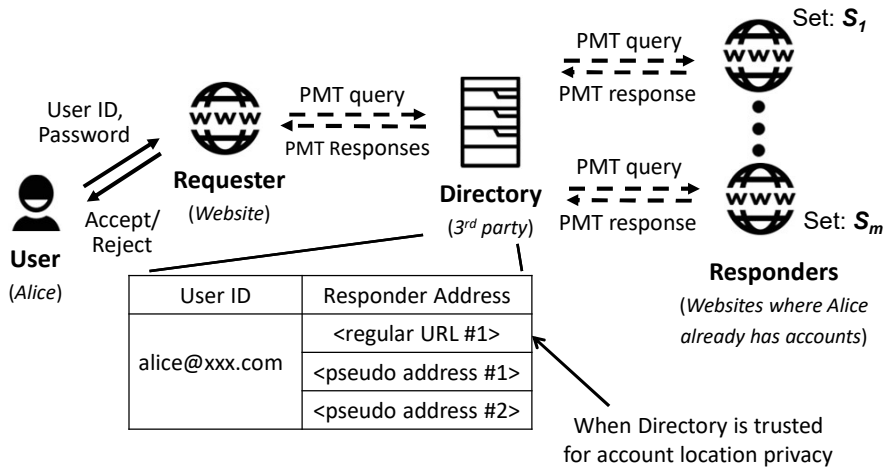
Account Location Privacy

35

35

Interfering with Password Reuse ::

Design :: Account Location Privacy



User ID	Responder Address
alice@xxx.com	<regular URL #1>
	<pseudo address #1>
	<pseudo address #2>

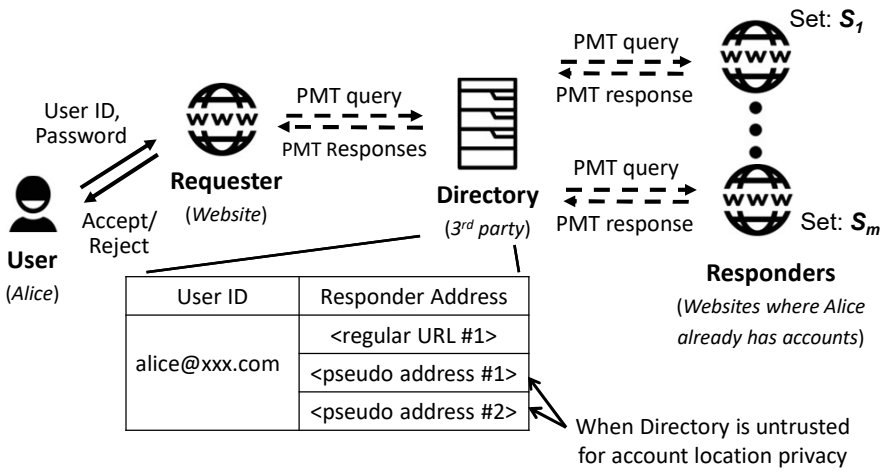
When Directory is trusted for account location privacy

36

36

Interfering with Password Reuse ::

Design :: Account Location Privacy



User ID	Responder Address
alice@xxx.com	<regular URL #1>
	<pseudo address #1>
	<pseudo address #2>

When Directory is untrusted for account location privacy

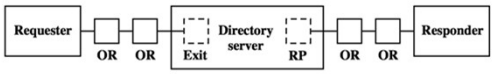
37

37

Interfering with Password Reuse ::

Design :: Account Location Privacy

Tor (The Onion Router) network enables anonymous communication, which can hide the identities of the **requester** and **responders** when the directory is **untrusted** for **account location privacy**.



A private Tor network for our prototype system, across 8 different datacenters in Europe and North America.

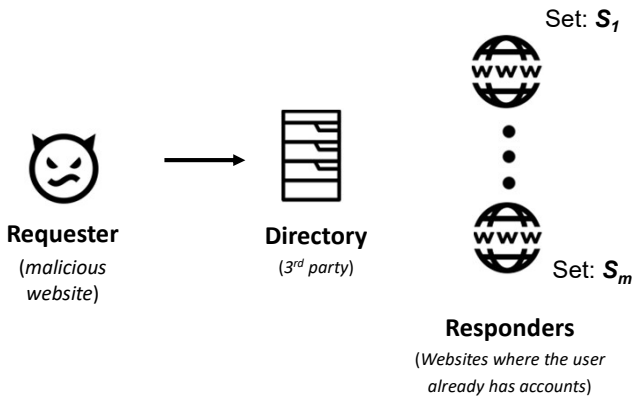
38

38

Interfering with Password Reuse ::

Design :: Limiting PMT Queries

- Directory can send the user a confirmation URL upon receiving queries from the requester and requires the user's confirmation to proceed with the protocol.



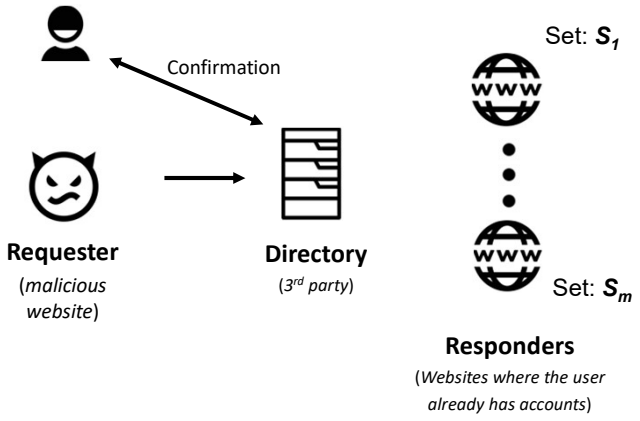
39

39

Interfering with Password Reuse ::

Design :: Limiting PMT Queries

- Directory can send the user a confirmation URL upon receiving queries from the requester and requires the user's confirmation to proceed with the protocol.



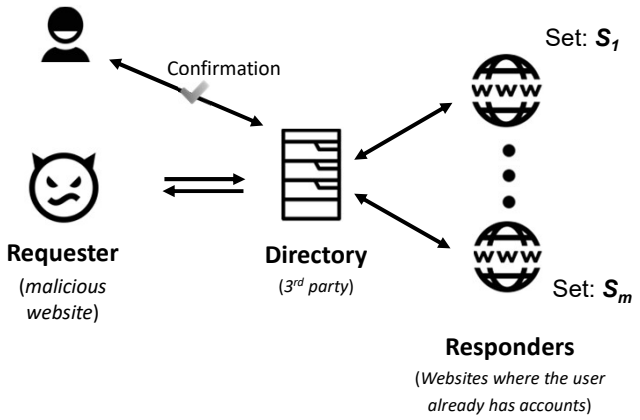
40

40

Interfering with Password Reuse ::

Design :: Limiting PMT Queries

- Directory can send the user a confirmation URL upon receiving queries from the requester and requires the user's confirmation to proceed with the protocol.

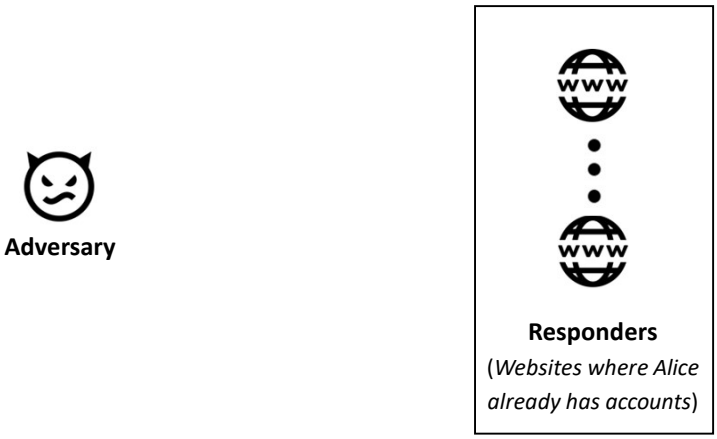


41

41

Interfering with Password Reuse ::

Probabilistic Model Checking



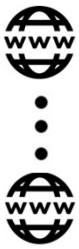
42

42

Interfering with Password Reuse ::

Probabilistic Model Checking

Adversary
(Markov Decision Process)



Responders
(*Websites where Alice already has accounts*)

43

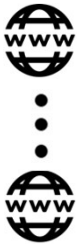
43

Interfering with Password Reuse ::

Probabilistic Model Checking

Prior knowledge about Alice's passwords (the "dictionary")

Adversary
(Markov Decision Process)



Responders
(*Websites where Alice already has accounts*)

44

44

Interfering with Password Reuse ::

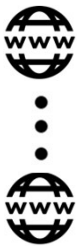
Probabilistic Model Checking

Prior knowledge about Alice's passwords (the "dictionary")

Adversary
(Markov Decision Process)

PMT
(with budgets)

Login
(with budgets)



Responders
(*Websites where Alice already has accounts*)

45

45

Interfering with Password Reuse ::

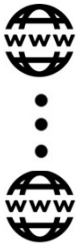
Probabilistic Model Checking

Prior knowledge about Alice's passwords (the "dictionary")

Adversary
(Markov Decision Process)

PMT
(with budgets)

Login
(with budgets)



Responders
(*Websites where Alice already has accounts*)

Adversary's goal: to compromise **at least one** account on those responders.

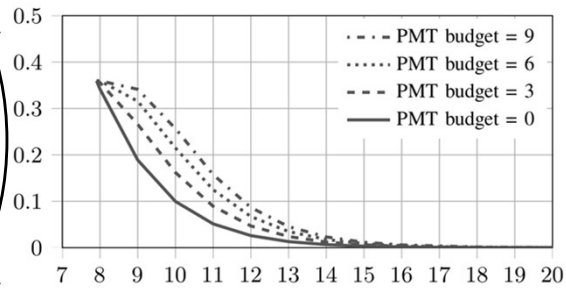
46

46

Interfering with Password Reuse ::

Probabilistic Model Checking

Max. Prob. of the adversary's success (at least one account takeover)



Dictionary size (bits)

# of websites: 12

Login budgets: 9

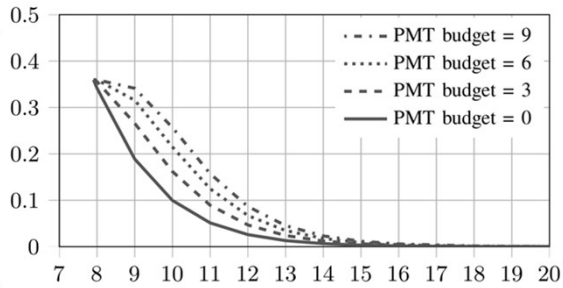
47

47

Interfering with Password Reuse ::

Probabilistic Model Checking

Max. Prob. of the adversary's success (at least one account takeover)



Dictionary size (bits)

Difficulty of guessing a user's passwords, given different levels of prior knowledge

# of websites: 12

Login budgets: 9

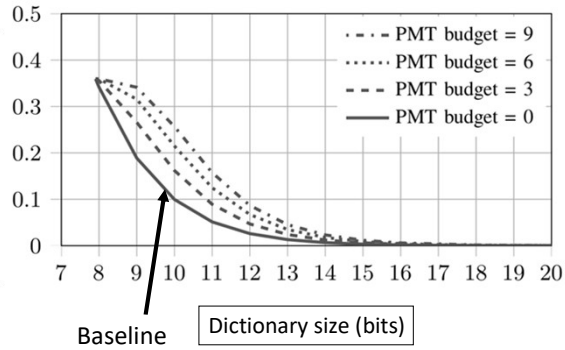
48

48

Interfering with Password Reuse ::

Probabilistic Model Checking

Max. Prob. of the adversary's success (at least one account takeover)



# of websites: 12

Login budgets: 9

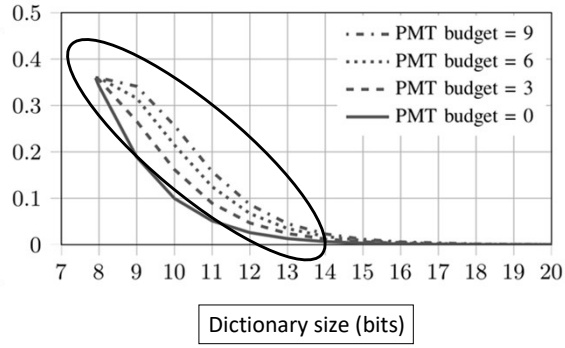
49

49

Interfering with Password Reuse ::

Probabilistic Model Checking

Max. Prob. of the adversary's success (at least one account takeover)



# of websites: 12

Login budgets: 9

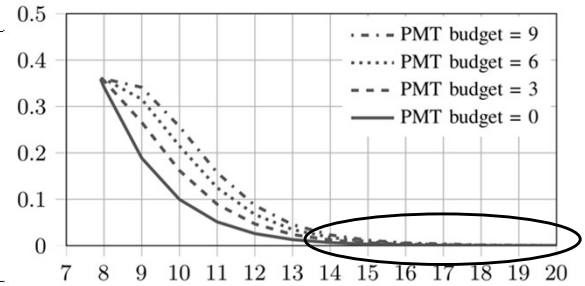
50

50

Interfering with Password Reuse ::

Probabilistic Model Checking

Max. Prob. of the adversary's success (at least one account takeover)



Dictionary size (bits)

# of websites: 12
Login budgets: 9

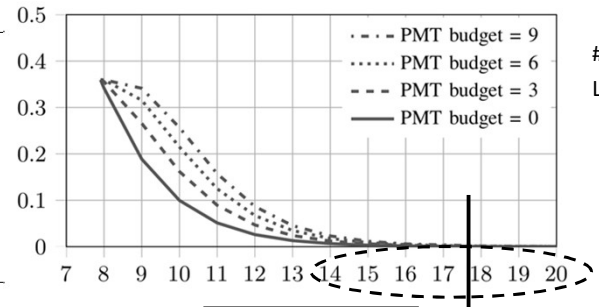
51

51

Interfering with Password Reuse ::

Probabilistic Model Checking

Max. Prob. of the adversary's success (at least one account takeover)



Dictionary size (bits)

# of websites: 12
Login budgets: 9

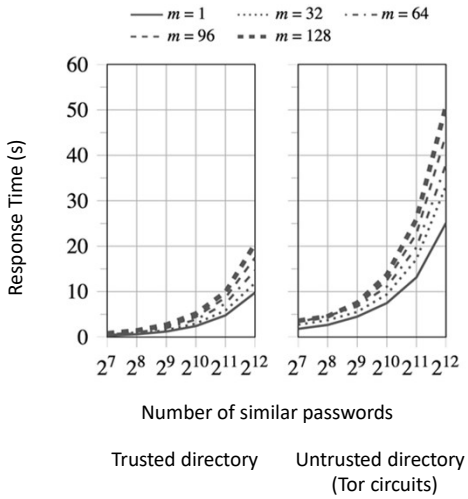
a password with only **3** randomly generated characters (a-z, A-Z, 0-9).

52

52

Interfering with Password Reuse ::

Performance :: Response Time



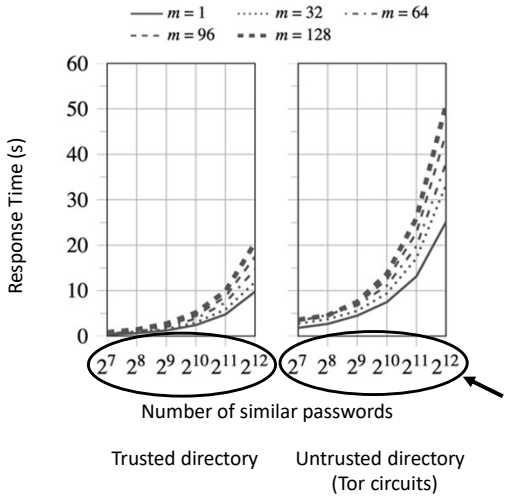
m: Number of responders

53

53

Interfering with Password Reuse ::

Performance :: Response Time



m: Number of responders

Does a responder actually need to deploy a password set this large?

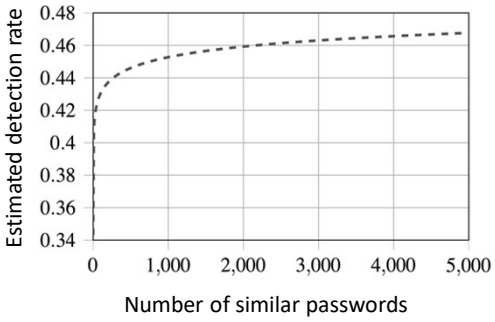
54

54

Interfering with Password Reuse ::

Detection Rate & Scalability

Prior Empirical Study about Password Reuse [1]



- An estimate of probability of detecting password reuse as a function of the number of similar passwords.
- Detection rate **increases sharply** when set of similar passwords is small. Adding to similar-password set **doesn't improve detection much, but it does increase overhead.**

[1] WANG, C., JAN, S. T. K., HU, H., AND WANG, G. Empirical analysis of password reuse and modification across online service. arXiv preprint arXiv:1706.01939 (2017)

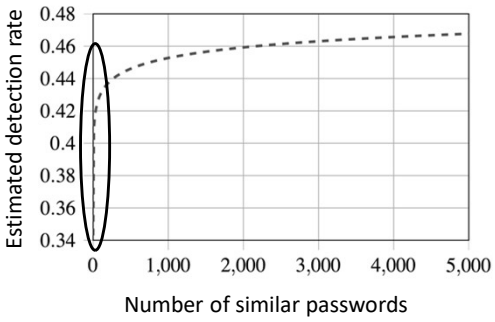
55

55

Interfering with Password Reuse ::

Detection Rate & Scalability

Prior Empirical Study about Password Reuse [1]



- An estimate of probability of detecting password reuse as a function of the number of similar passwords.
- Detection rate **increases sharply** when set of similar passwords is small. Adding to similar-password set **doesn't improve detection much, but it does increase overhead.**

[1] WANG, C., JAN, S. T. K., HU, H., AND WANG, G. Empirical analysis of password reuse and modification across online service. arXiv preprint arXiv:1706.01939 (2017)

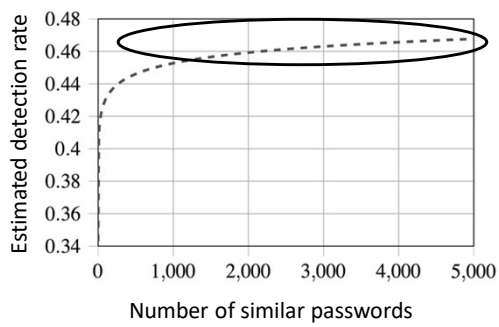
56

56

Interfering with Password Reuse ::

Detection Rate & Scalability

Prior Empirical Study about Password Reuse [1]



- An estimate of probability of detecting password reuse as a function of the number of similar passwords.
- Detection rate **increases sharply** when set of similar passwords is small. Adding to similar-password set **doesn't improve detection much, but it does increase overhead.**

[1] WANG, C., JAN, S. T. K., HU, H., AND WANG, G. Empirical analysis of password reuse and modification across online service. arXiv preprint arXiv:1706.01939 (2017)

57

57

Interfering with Password Reuse ::

Detection Rate & Scalability

True detection rate maximization

Given a **target response time** constraint, how to choose **number of similar passwords (*n*)** and **number of participating responders (*m*)** to maximize **true detection rate**

	t_{goal} (s)									
	.01	.02	.03	.04	.05	.06	.07	.08	.09	.10
n	1	1	2	2	5	9	13	16	20	23
m	1	10	17	26	26	26	26	26	26	26
tdr	.343	.985	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1

Trusted directory

	t_{goal} (s)									
	1.60	1.62	1.64	1.66	1.68	1.70	1.72	1.74	1.76	1.78
n	1	2	2	5	8	11	14	17	19	22
m	16	21	26	26	26	26	26	26	26	26
tdr	.999	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1

Untrusted directory

t_{goal}: target response time;
m: number of responders;
n: number of similar passwords;
tdr: overall true detection rate.

Max qualifying responses per sec.

n	*m* = 1	6	11	16	21	26
1	4304	1013	492	325	237	174
10	2415	549	277	188	155	122
20	1478	336	182	129	98	78
30	1076	243	124	86	63	53
40	788	187	94	67	49	40
50	683	159	76	52	39	33
60	611	132	63	43	32	25

Trusted directory
(Qualifying response: <= 5s)

n	*m* = 1	6	11	16	21	26
1	95	61	42	33	27	22
10	87	59	40	31	25	20
20	78	54	37	28	23	19
30	71	51	35	27	20	16
40	62	44	32	24	18	14
50	53	39	26	20	15	11
60	42	31	20	16	10	10

Untrusted directory
(Qualifying response: <= 8s)

58

58

Interfering with Password Reuse ::

Detection Rate & Scalability

True detection rate maximization

Given a **target response time** constraint, how to choose **number of similar passwords (*n*)** and **number of participating responders (*m*)** to maximize **true detection rate**

	t_{goal} (s)									
	.01	.02	.03	.04	.05	.06	.07	.08	.09	.10
n	1	1	2	2	5	9	13	16	20	23
m	1	10	17	26	26	26	26	26	26	26
tdr	.343	.985	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1

Trusted directory

	t_{goal} (s)									
	1.60	1.62	1.64	1.66	1.68	1.70	1.72	1.74	1.76	1.78
n	1	2	2	5	8	11	14	17	19	22
m	16	21	26	26	26	26	26	26	26	26
tdr	.999	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1

Untrusted directory

t_{goal}: target response time;
m: number of responders;
n: number of similar passwords;
tdr: overall true detection rate.

Max qualifying responses per sec.

n \ *m*	1	6	11	16	21	26
1	4304	1013	492	325	237	174
10	2415	549	277	188	155	122
20	1478	336	182	129	98	78
30	1076	243	124	86	63	53
40	788	187	94	67	49	40
50	683	159	76	52	39	33
60	611	132	63	43	32	25

Trusted directory
(Qualifying response: <= 5s)

n \ *m*	1	6	11	16	21	26
1	95	61	42	33	27	22
10	87	59	40	31	25	20
20	78	54	37	28	23	19
30	71	51	35	27	20	16
40	62	44	32	24	18	14
50	53	39	26	20	15	11
60	42	31	20	16	10	10

Untrusted directory
(Qualifying response: <= 8s)

59

59

Interfering with Password Reuse ::

Detection Rate & Scalability

True detection rate maximization

Given a **target response time** constraint, how to choose **number of similar passwords (*n*)** and **number of participating responders (*m*)** to maximize **true detection rate**

	t_{goal} (s)									
	.01	.02	.03	.04	.05	.06	.07	.08	.09	.10
n	1	1	2	2	5	9	13	16	20	23
m	1	10	17	26	26	26	26	26	26	26
tdr	.343	.985	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1

Trusted directory

	t_{goal} (s)									
	1.60	1.62	1.64	1.66	1.68	1.70	1.72	1.74	1.76	1.78
n	1	2	2	5	8	11	14	17	19	22
m	16	21	26	26	26	26	26	26	26	26
tdr	.999	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1

Untrusted directory

t_{goal}: target response time;
m: number of responders;
n: number of similar passwords;
tdr: overall true detection rate.

Max qualifying responses per sec.

n \ *m*	1	6	11	16	21	26
1	4304	1013	492	325	237	174
10	2415	549	277	188	155	122
20	1478	336	182	129	98	78
30	1076	243	124	86	63	53
40	788	187	94	67	49	40
50	683	159	76	52	39	33
60	611	132	63	43	32	25

Trusted directory
(Qualifying response: <= 5s)

n \ *m*	1	6	11	16	21	26
1	95	61	42	33	27	22
10	87	59	40	31	25	20
20	78	54	37	28	23	19
30	71	51	35	27	20	16
40	62	44	32	24	18	14
50	53	39	26	20	15	11
60	42	31	20	16	10	10

Untrusted directory
(Qualifying response: <= 8s)

Number of responders

Number of similar passwords (or set size) at responders

60

60

Interfering with Password Reuse ::

Detection Rate & Scalability

True detection rate maximization

Given a **target response time** constraint, how to choose **number of similar passwords (*n*)** and **number of participating responders (*m*)** to maximize **true detection rate**

	t_{goal} (s)									
	.01	.02	.03	.04	.05	.06	.07	.08	.09	.10
n	1	1	2	2	5	9	13	16	20	23
m	1	10	17	26	26	26	26	26	26	26
tdr	.343	.985	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1

Trusted directory

	t_{goal} (s)									
	1.60	1.62	1.64	1.66	1.68	1.70	1.72	1.74	1.76	1.78
n	1	2	2	5	8	11	14	17	19	22
m	16	21	26	26	26	26	26	26	26	26
tdr	.999	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1

Untrusted directory

t_{goal}: target response time;
m: number of responders;
n: number of similar passwords;
tdr: overall true detection rate.

Max qualifying responses per sec.

n \ *m*	1	6	11	16	21	26
1	4304	1013	492	325	237	174
10	2415	549	277	188	155	122
20	1478	336	182	129	98	78
30	1076	243	124	86	63	53
40	788	187	94	67	49	40
50	683	159	76	52	39	33
60	611	132	63	43	32	25

Trusted directory
(Qualifying response: <= 5s)

n \ *m*	1	6	11	16	21	26
1	95	61	42	33	27	22
10	87	59	40	31	25	20
20	78	54	37	28	23	19
30	71	51	35	27	20	16
40	62	44	32	24	18	14
50	53	39	26	20	15	11
60	42	31	20	16	10	10

Untrusted directory
(Qualifying response: <= 8s)

61

61

Interfering with Password Reuse ::

Detection Rate & Scalability

True detection rate maximization

Given a **target response time** constraint, how to choose **number of similar passwords (*n*)** and **number of participating responders (*m*)** to

tdr .343 .985 ≈1 ≈1 ≈1 ≈1 ≈1 ≈1 ≈1 ≈1

Trusted directory

	t_{goal} (s)									
	1.60	1.62	1.64	1.66	1.68	1.70	1.72	1.74	1.76	1.78
n	1	2	2	5	8	11	14	17	19	22
m	16	21	26	26	26	26	26	26	26	26
tdr	.999	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1	≈1

Untrusted directory

t_{goal}: target response time;
m: number of responders;
n: number of similar password;
tdr: overall true detection rate.

Max qualifying responses per sec.

n \ *m*	1	6	11	16	21	26
1	4304	1013	492	325	237	174
10	2415	549	277	188	155	122

(Qualifying response: <= 5s)

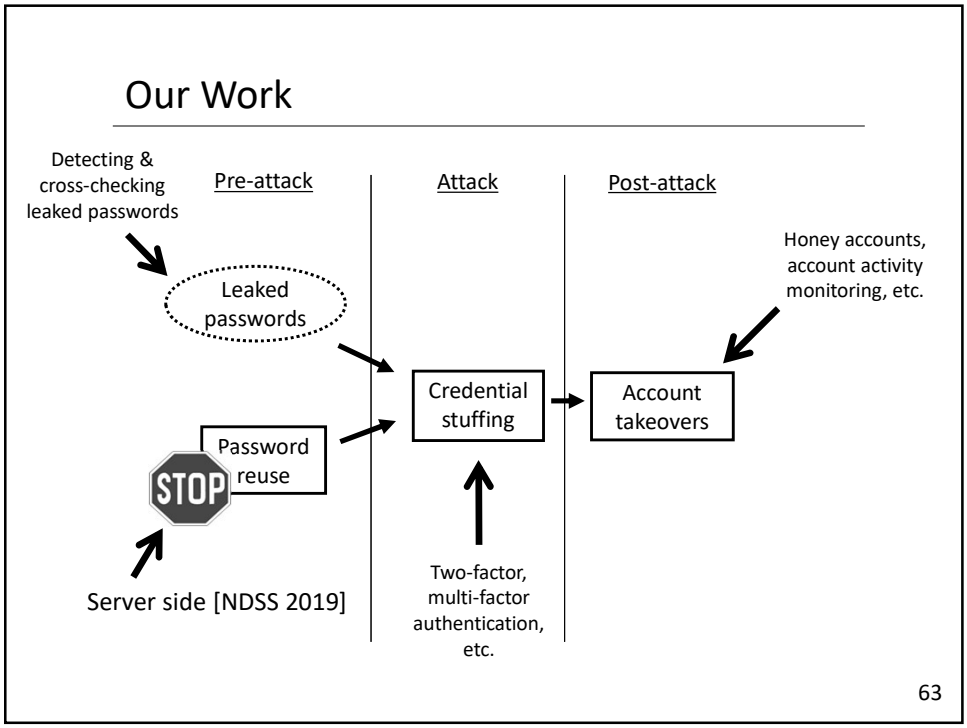
n \ *m*	1	6	11	16	21	26
1	95	61	42	33	27	22
10	87	59	40	31	25	20
20	78	54	37	28	23	19
30	71	51	35	27	20	16
40	62	44	32	24	18	14
50	53	39	26	20	15	11
60	42	31	20	16	10	10

Untrusted directory
(Qualifying response: <= 8s)

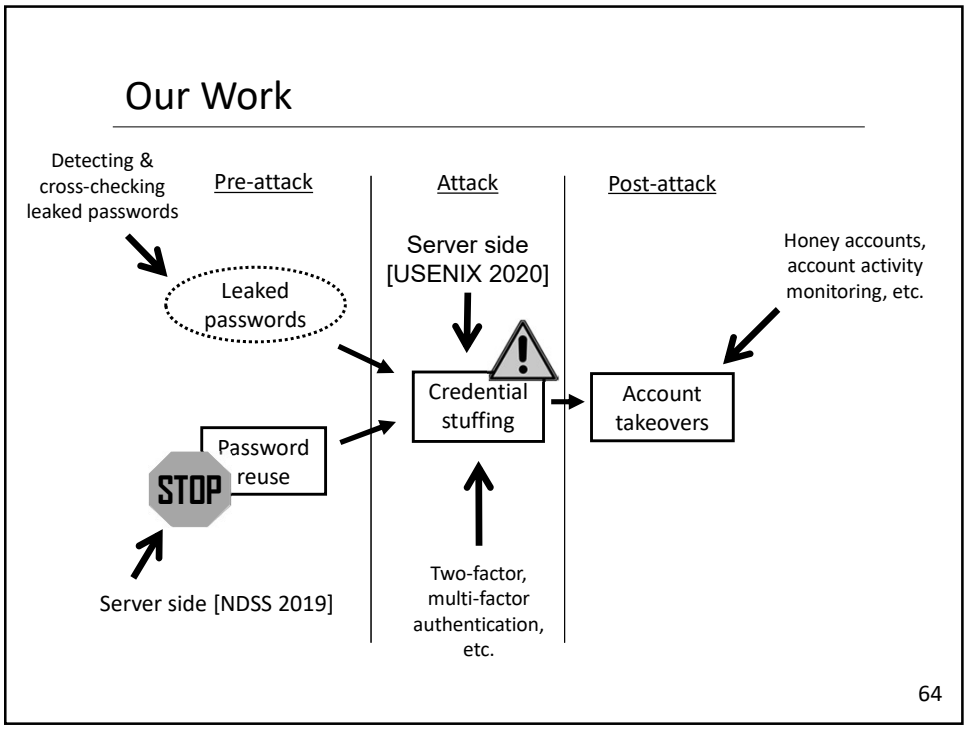
62

A quick estimate:
A throughput of 50 qualifying responses per second is enough to enable each of the about $3x10^8$ Internet users in the U.S. to setup or change passwords on more than 5 accounts per year.

62


Our Work
Detecting & cross-checking leaked passwords
Pre-attack
Attack
Post-attack
Honey accounts, account activity monitoring, etc.
Leaked passwords
Credential stuffing
Account takeovers
Password reuse
STOP
Server side [NDSS 2019]
Two-factor, multi-factor authentication, etc.
63

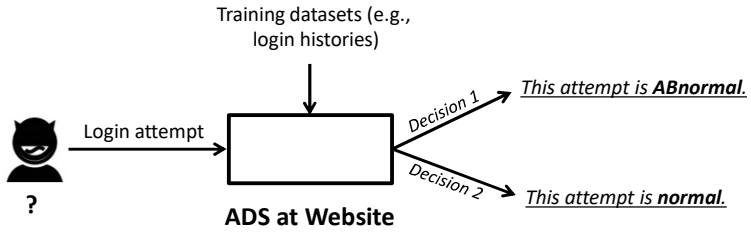
63


Our Work
Detecting & cross-checking leaked passwords
Pre-attack
Attack
Post-attack
Server side [USENIX 2020]
Honey accounts, account activity monitoring, etc.
Leaked passwords
Credential stuffing
Account takeovers
Password reuse
STOP
Server side [NDSS 2019]
Two-factor, multi-factor authentication, etc.
64

64

Detecting Credential Stuffing ::

Anomaly Detection Systems (ADS)

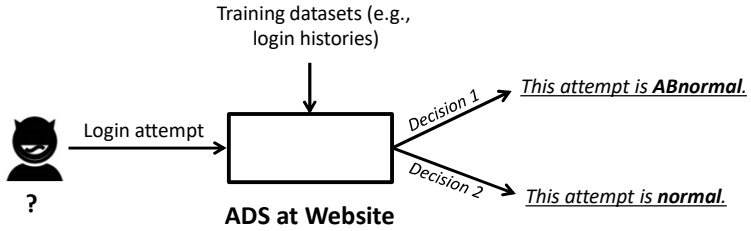


65

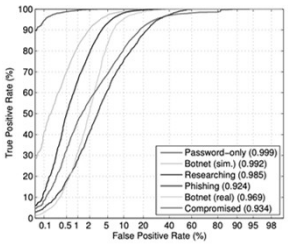
65

Detecting Credential Stuffing ::

Anomaly Detection Systems (ADS)



- Login features: IP addresses, useragent strings, device fingerprints, etc.
- Naïve ADS:
  - Strange IPs = *"abnormal"*
  - Strange devices = *"abnormal"*
- A proposed ADS*
  - Attacker capabilities
  - Classifier thresholds



* D. Freeman, S. Jain, M. Dürmuth, B. Biggio, and G. Giacinto. Who are you? A statistical approach to measuring user authenticity. In 23rd ISOC NDSS, February 2016.

66

66

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts

67

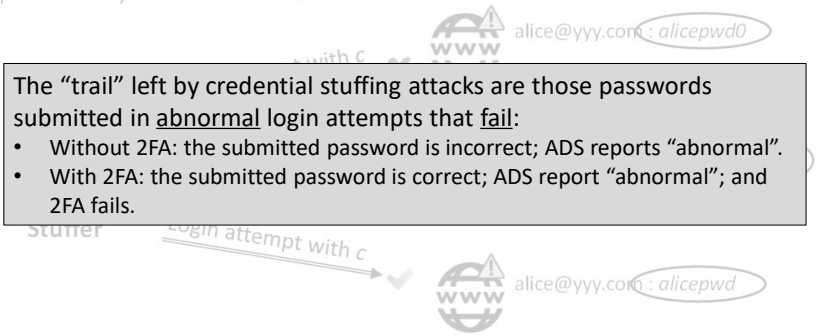
67

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts



The "trail" left by credential stuffing attacks are those passwords submitted in abnormal login attempts that fail:

- Without 2FA: the submitted password is incorrect; ADS reports "abnormal".
- With 2FA: the submitted password is correct; ADS report "abnormal"; and 2FA fails.

68

68

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts

alice@yyy.com : *alicepwd0*

alice@yyy.com : *alicepwd* + 2FA

alice@yyy.com : *alicepwd*

Credential Stuffer

69

69

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts

alice@yyy.com : *alicepwd0*

Suspicious: { alicepwd }

Login attempt with *c*

alice@yyy.com : *alicepwd* + 2FA

alice@yyy.com : *alicepwd*

Credential Stuffer

70

70

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts

alice@yyy.com : *alicepwd0*
Suspicious: { alicepwd }

Credential Stuffer

Login attempt with *c*

alice@yyy.com : *alicepwd* + 2FA
Suspicious: { alicepwd }

alice@yyy.com : *alicepwd*

71

71

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts

alice@yyy.com : *alicepwd0*
Suspicious: { alicepwd }

Collecting phase

Credential Stuffer

alice@yyy.com : *alicepwd* + 2FA
Suspicious: { alicepwd }

alice@yyy.com : *alicepwd*

72

72

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts

alice@yyy.com : *alicepwd0*
Suspicious: { alicepwd }

Credential Stuffer

alice@yyy.com : *alicepwd* + 2FA
Suspicious: { alicepwd }

Login attempt with *c*

alice@yyy.com : *alicepwd*

73

73

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts

alice@yyy.com : *alicepwd0*
Suspicious: { alicepwd }

Credential Stuffer

Do you have "alicepwd"
as suspicious?

alice@yyy.com : *alicepwd* + 2FA
Suspicious: { alicepwd }

Login attempt with *c*

alice@yyy.com : *alicepwd*

74

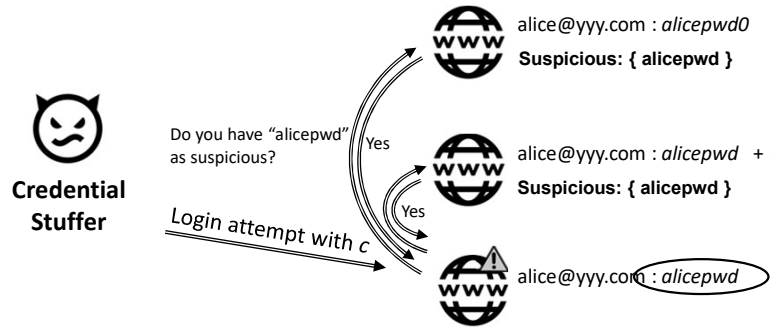
74

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts



A positive detection happens when the number of positives is at least a pre-set threshold ("**attack width**").

75

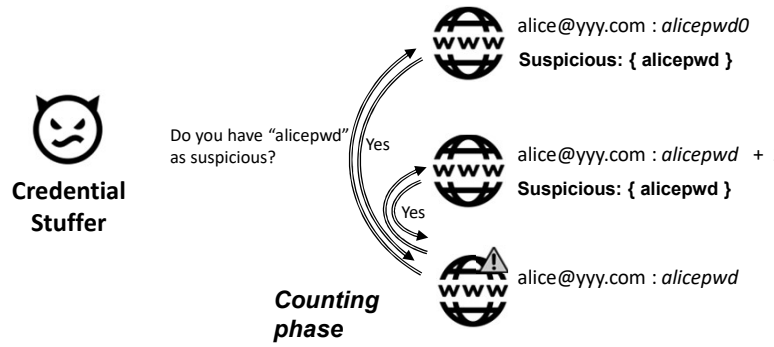
75

Detecting Credential Stuffing ::

Evidence Trail from Credential Stuffing

c = "alice@yyy.com : ***alicepwd***",
a leaked username-password pair
possessed by the credential stuffer

Websites where Alice
has accounts



A positive detection happens when the number of positives is at least a pre-set threshold ("**attack width**").

76

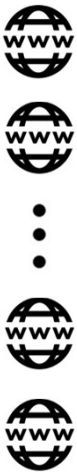
76

Detecting Credential Stuffing ::

Estimating False Detection Rate

Trying to recall **password-account mappings** by credential stuffing his/her own accounts with password candidates

Forgetful User (Mr. Hyde)



Websites

77

77

Detecting Credential Stuffing ::

Estimating False Detection Rate

Trying to recall **password-account mappings** by credential stuffing his/her own accounts with password candidates

Forgetful User (Mr. Hyde)

Try to log in with password *p, p',…*

Try to log in with password *p, p', …*

Try to log in with password *p, p', …*

Try to log in with password *p, p', …*

Websites

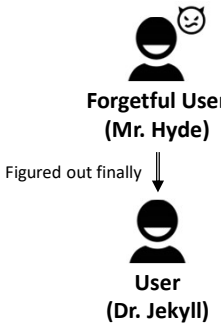
78

78

Detecting Credential Stuffing ::

Estimating False Detection Rate

Trying to recall **password-account mappings** by credential stuffing his/her own accounts with password candidates



Websites

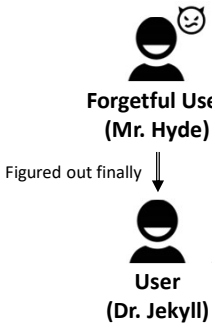
79

79

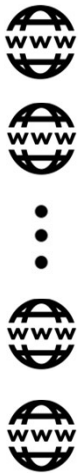
Detecting Credential Stuffing ::

Estimating False Detection Rate

Trying to recall **password-account mappings** by credential stuffing his/her own accounts with password candidates



Log in an account with a correct password



Websites

80

80

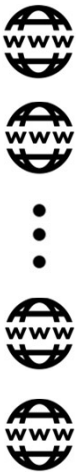
Detecting Credential Stuffing ::

Estimating False Detection Rate :: MDP

With knowledge of:
- All valid passwords of the user
- Password distribution (frequency)
- All sites where the user has accounts
- Credentials to pass any 2FA

Without knowledge of:
- Mapping between passwords and site(s) where they are correct

MDP
(Mr. Hyde)



Websites

* MDP: Markov decision process

81

81

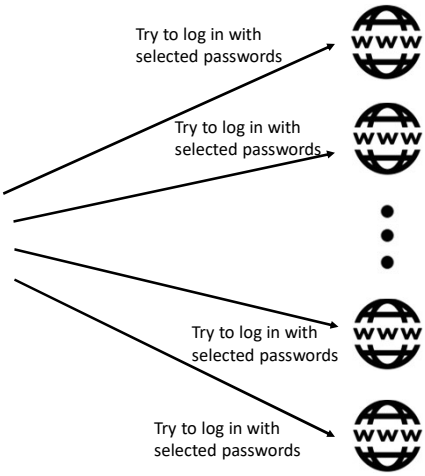
Detecting Credential Stuffing ::

Estimating False Detection Rate :: MDP

With knowledge of:
- All valid passwords of the user
- Password distribution (frequency)
- All sites where the user has accounts
- Credentials to pass any 2FA

Without knowledge of:
- Mapping between passwords and site(s) where they are correct

MDP
(Mr. Hyde)



Websites

* MDP: Markov decision process

82

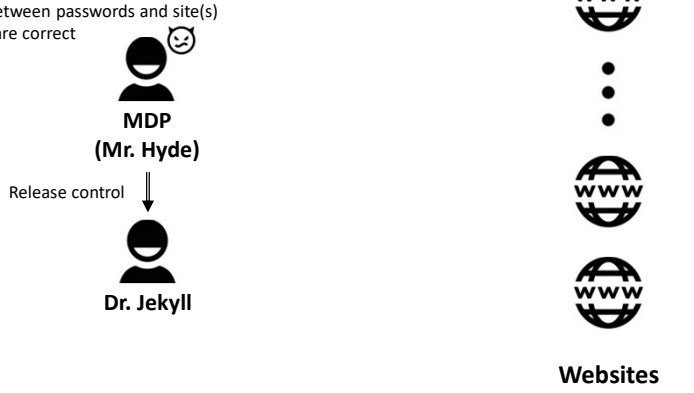
82

Detecting Credential Stuffing ::

Estimating False Detection Rate :: MDP

With knowledge of:
- All valid passwords of the user
- Password distribution (frequency)
- All sites where the user has accounts
- Credentials to pass any 2FA

Without knowledge of:
- Mapping between passwords and site(s) where they are correct



* MDP: Markov decision process

83

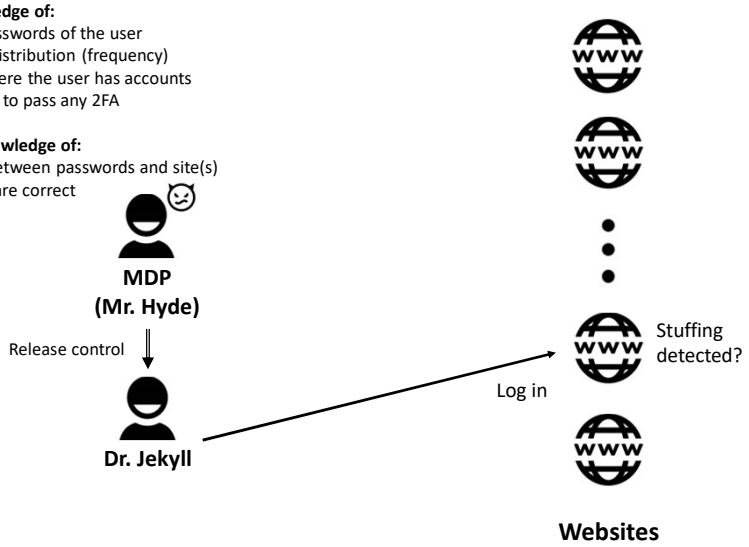
83

Detecting Credential Stuffing ::

Estimating False Detection Rate :: MDP

With knowledge of:
- All valid passwords of the user
- Password distribution (frequency)
- All sites where the user has accounts
- Credentials to pass any 2FA

Without knowledge of:
- Mapping between passwords and site(s) where they are correct



* MDP: Markov decision process

84

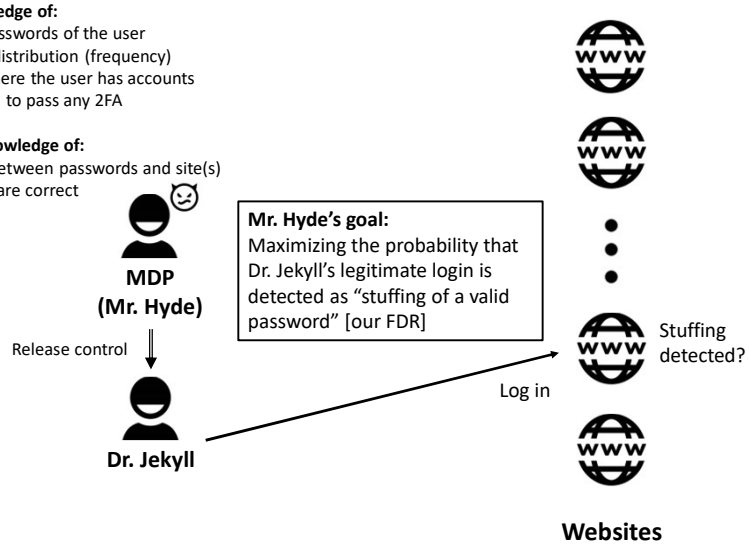
84

Detecting Credential Stuffing ::

Estimating False Detection Rate :: MDP

With knowledge of:
- All valid passwords of the user
- Password distribution (frequency)
- All sites where the user has accounts
- Credentials to pass any 2FA

Without knowledge of:
- Mapping between passwords and site(s) where they are correct



Mr. Hyde's goal:
Maximizing the probability that Dr. Jekyll's legitimate login is detected as "stuffing of a valid password" [our FDR]

* MDP: Markov decision process

85

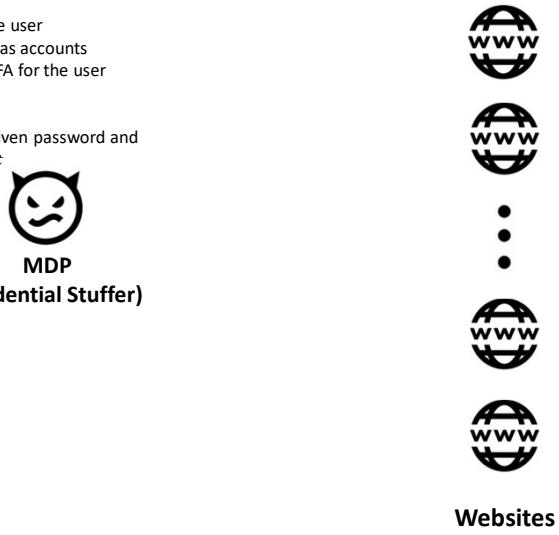
85

Detecting Credential Stuffing ::

Estimating True Detection Rate :: MDP

With knowledge of:
- One valid password of the user
- All sites where the user has accounts
- Which site(s) deployed 2FA for the user

Without knowledge of:
- Mappings between the given password and the sites where it is correct
- Credentials to pass 2FA



* MDP: Markov decision process

86

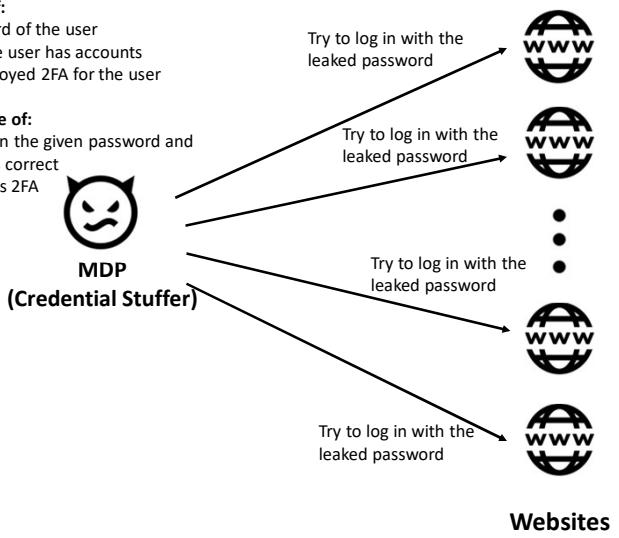
86

Detecting Credential Stuffing ::

Estimating True Detection Rate :: MDP

With knowledge of:
- One valid password of the user
- All sites where the user has accounts
- Which site(s) deployed 2FA for the user

Without knowledge of:
- Mappings between the given password and the sites where it is correct
- Credentials to pass 2FA



* MDP: Markov decision process

87

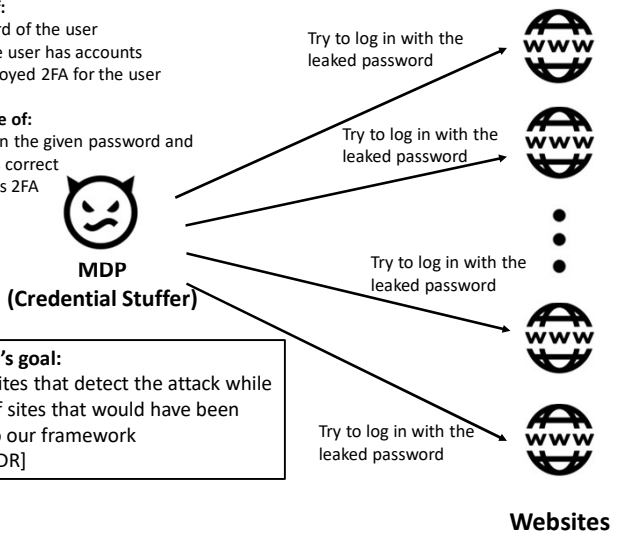
87

Detecting Credential Stuffing ::

Estimating True Detection Rate :: MDP

With knowledge of:
- One valid password of the user
- All sites where the user has accounts
- Which site(s) deployed 2FA for the user

Without knowledge of:
- Mappings between the given password and the sites where it is correct
- Credentials to pass 2FA



Credential Stuffer's goal:
Minimize E(#) of sites that detect the attack while maximizing E(#) of sites that would have been compromised w/o our framework
[The ratio is our TDR]

* MDP: Markov decision process

88

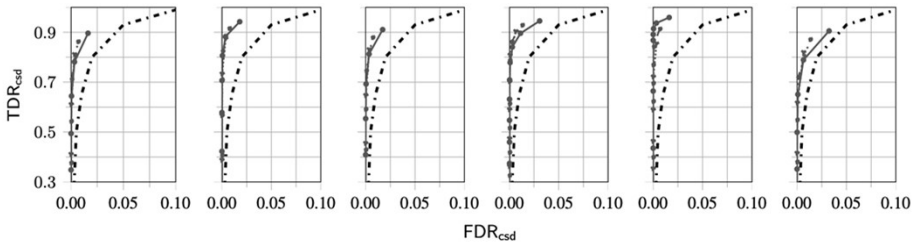
88

Detecting Credential Stuffing ::

Trading Off FDR & TDR

Researching attackers[5]: valid passwords from same countries as legitimiate users.

- **Default (baseline) setting**: some level of password reuse in a set of 4 distinct passwords across 10 accounts/sites with no 2FA deployed among them
- **Blue curves**: each for a different ADS threshold in *the collecting phase*
- **Black curves**: corresponding ADS's accuracy in detecting suspicious logins



Baseline | Less pwd reuse | # of pwds +1 | # of accnts + 10 | # of 2FA + 5 | Higher ADS detection rates in the *counting phase*

[5] Freeman et al. (NDSS 2016)

89

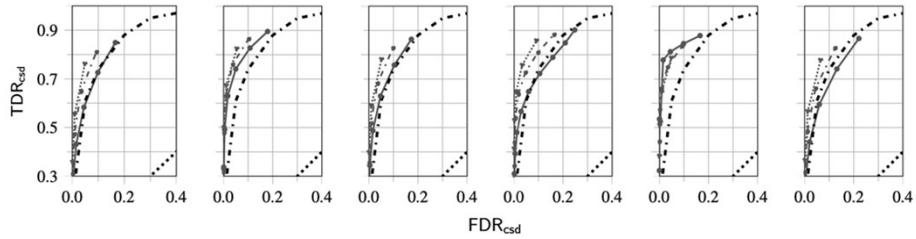
89

Detecting Credential Stuffing ::

Trading Off FDR & TDR

Phishing attackers[5]: valid passwords from same countries with same browser user-agent strings of legitimate users

- **Default (baseline) setting**: some level of password reuse in a set of 4 distinct passwords across 10 accounts/sites with no 2FA deployed among them
- **Blue curves**: each for a different ADS threshold in *the collecting phase*
- **Black, dashed curves**: corresponding ADS's accuracy in detecting suspicious logins
- **Black, dotted lines**: random guessing



Baseline | Less pwd reuse | # of pwds +1 | # of accnts + 10 | # of 2FA + 5 | Higher ADS detection rates in the *counting phase*

[5] Freeman et al. (NDSS 2016)

90

90

Detecting Credential Stuffing ::

Goals :: Security and Privacy

- **Account location privacy**: Participating websites are not disclosed to one another

91

91

Detecting Credential Stuffing ::

Goals :: Security and Privacy

- ~~**Account location privacy**: Participating websites are not disclosed to one another~~
- **Login privacy**: Hide ***where the user is currently trying to log in*** from other participating websites

92

92

Detecting Credential Stuffing ::

Goals :: Security and Privacy

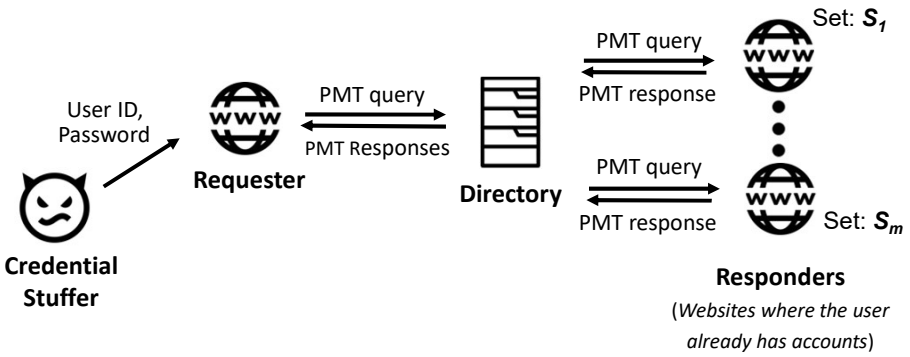
- ~~**Account location privacy**: Participating websites are not disclosed to one another~~
- **Login privacy**: Hide ***where the user is currently trying to log in*** from other participating websites
- **Account security**: Detect credential stuffing while not qualitatively degrading account security in other ways

93

93

Detecting Credential Stuffing ::

Framework :: Directory



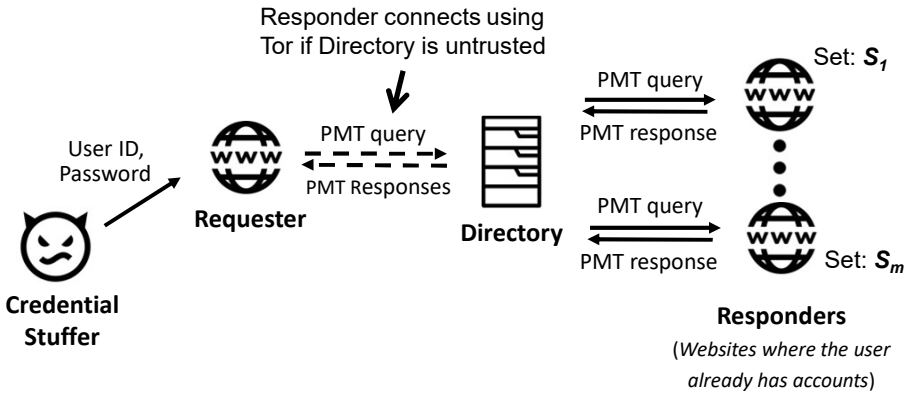
S_i: Suspicious-password-set for the user at responder *i*

94

94

Detecting Credential Stuffing ::

Login Privacy



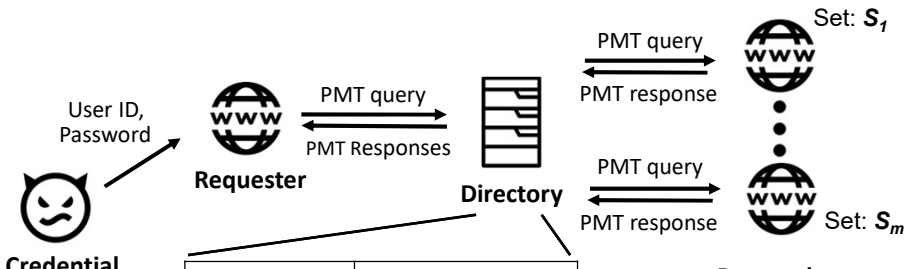
S_i : Suspicious-password-set for the user at responder *i*

95

95

Detecting Credential Stuffing ::

Login Privacy



User ID	Responder Address
alice@xxx.com	<regular URL #1>
	…
	<regular URL #*m*>

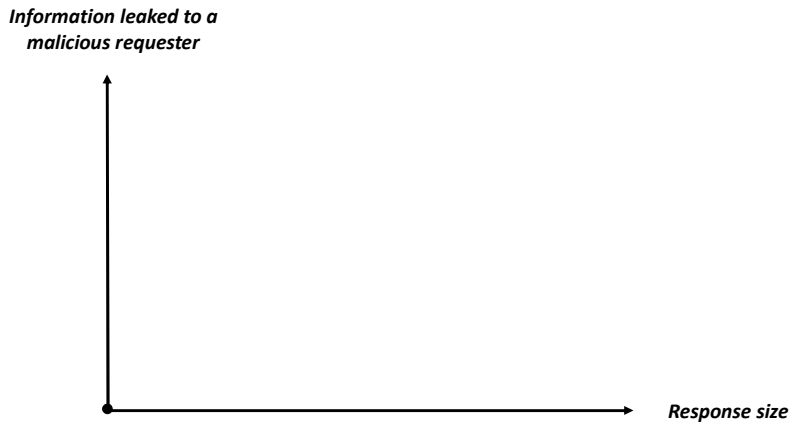
S_i : Suspicious-password-set for the user at responder *i*

96

96

Detecting Credential Stuffing ::

PMTs :: One-Round Protocols

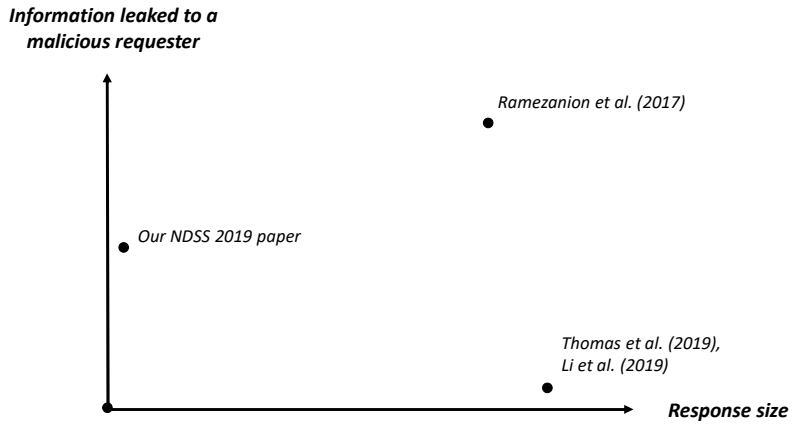


97

97

Detecting Credential Stuffing ::

PMTs :: One-Round Protocols

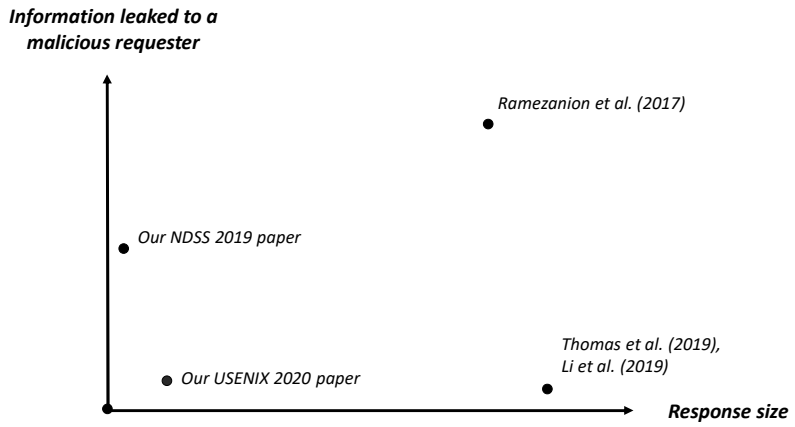


98

98

Detecting Credential Stuffing ::

PMTs :: One-Round Protocols



99

99

Detecting Credential Stuffing ::

Our PMT :: Cuckoo Filters

- An approximate membership query (AMQ) scheme proposed by Fan et al. (2014)
- Time needed by a membership test is constant with respect to the set size
- Space-efficient storage with tunable false positive rates
  - More space-efficient than Bloom filters when false positive rates are low (e.g., < 3%)

100

100

Detecting Credential Stuffing ::

Our PMT :: Cuckoo Filters :: Membership Test

A fingerprint function: fp()
Two hash functions: $h_1()$ = hash()
$h_2()$ = hash() $\oplus$ hash(fp())

101

101

Detecting Credential Stuffing ::

Our PMT :: Cuckoo Filters :: Membership Test

A fingerprint function: fp()
Two hash functions: $h_1()$ = hash()
$h_2()$ = hash() $\oplus$ hash(fp())

Bucket **1**	Bucket **2**	Bucket **3**	Bucket **4**	Bucket **5**	Bucket **6**
	fp("eve")				fp("bob")
	fp("charlie")		fp("alice")		fp("frank")

h_1("alice") = **2** ? ? h_2("alice") = **4**

Test membership of "alice"

102

102

Detecting Credential Stuffing ::

Our PMT :: Cuckoo Filters :: Membership Test

A fingerprint function: fp()
Two hash functions: $h_1()$ = hash()
$h_2()$ = hash() $\oplus$ hash(fp())

Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6
	fp("eve")				fp("bob")
	fp("charlie")		fp("alice")		fp("frank")

h_1("alice") = 2 ✗ ✓ h_2("alice") = 4

Test membership of "alice"

If at least one slot in the corresponding two buckets contains the queried fingerprint, membership holds.

103

103

Detecting Credential Stuffing ::

Our PMT :: Cuckoo Filters :: Membership Test

A fingerprint function: fp()
Two hash functions: $h_1()$ = hash()
$h_2()$ = hash() $\oplus$ hash(fp())

Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6
	fp("eve")				fp("bob")
	fp("charlie")		fp("ice")		fp("frank")

h_1("alice") = **2** ? ? h_2("alice") = **4**

Test membership of "alice"

104

104

Detecting Credential Stuffing ::

Our PMT :: Cuckoo Filters :: Membership Test

A fingerprint function: fp()
Two hash functions: $h_1()$ = hash()
$h_2()$ = hash() $\oplus$ hash(fp())

Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5	Bucket 6
	fp("eve")				fp("bob")
	fp("charlie")		fp("ice")		fp("frank")

h_1("alice") = 2 ✗ ✗ h_2("alice") = 4

Test membership of "alice"

If none of the slots in the corresponding two buckets contains the queried fingerprint, membership does not hold.

105

105

Detecting Credential Stuffing ::

Our PMT :: Encryption

Additively Homomorphic Encryption

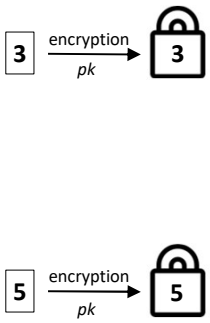
106

106

Detecting Credential Stuffing ::

Our PMT :: Encryption

Additively Homomorphic Encryption



$+_{pk}$: homomorphic addition (only *pk* is needed)
pk : public key (or "encryption key")
sk : private key (or "decryption key")

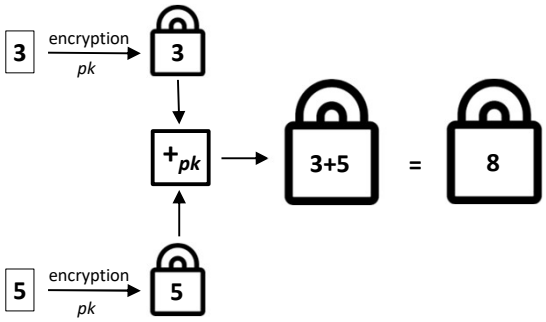
107

107

Detecting Credential Stuffing ::

Our PMT :: Encryption

Additively Homomorphic Encryption



$+_{pk}$: homomorphic addition (only *pk* is needed)
pk : public key (or "encryption key")
sk : private key (or "decryption key")

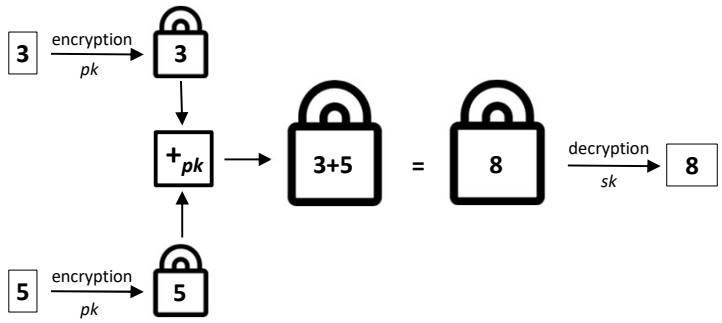
108

108

Detecting Credential Stuffing ::

Our PMT :: Encryption

Additively Homomorphic Encryption



$+_{pk}$: homomorphic addition (only *pk* is needed)
pk : public key (or "encryption key")
sk : private key (or "decryption key")

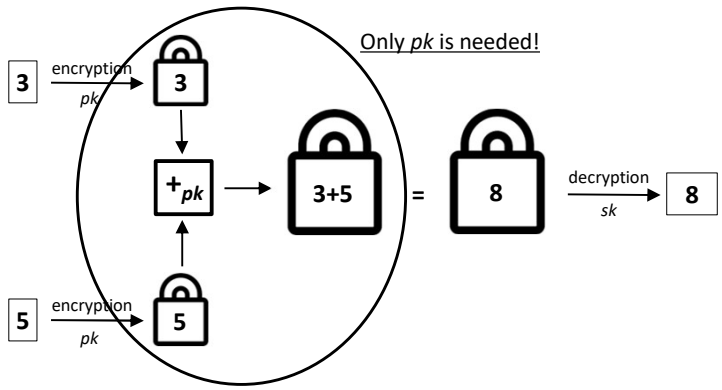
109

109

Detecting Credential Stuffing ::

Our PMT :: Encryption

Additively Homomorphic Encryption



$+_{pk}$: homomorphic addition (only *pk* is needed)
pk : public key (or "encryption key")
sk : private key (or "decryption key")

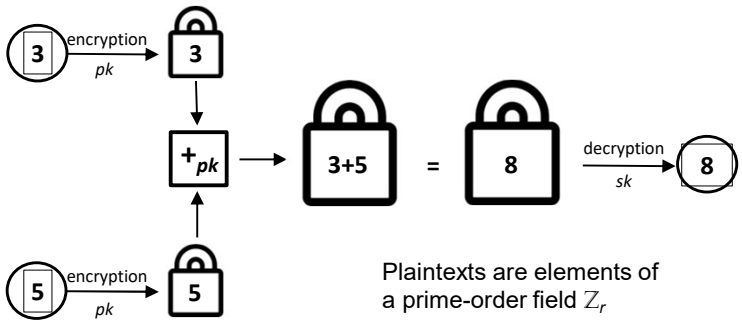
110

110

Detecting Credential Stuffing ::

Our PMT :: Encryption

Additively Homomorphic Encryption



Plaintexts are elements of a prime-order field $\mathbb{Z}_r$

$+_{pk}$: homomorphic addition (only *pk* is needed)
pk : public key (or "encryption key")
sk : private key (or "decryption key")

111

111

Detecting Credential Stuffing ::

Our PMT :: Construction

Requester
(Element: ***"bigbang"***, *pk, sk*)

Responder
(Set: ***S***)

Set ***S*** = {"***bigbang***", "***domingo***", … }

112

112

Detecting Credential Stuffing ::

Our PMT :: Construction

Requester
(Element: ***"bigbang"***, *pk, sk*)

Responder
(Set: ***S***)

Set ***S*** = {"***bigbang***", "***domingo***", … }

"bigbang" $\xrightarrow{h_1(),\ h_2()}$

0	***1***
0	0
1	0
0	0

fp()

– f = – fp(*"bigbang"*)

S $\xrightarrow[\text{fp()}]{h_1(),\ h_2()}$

fp_{1A}	fp_{2A}	fp_{3A}	fp_{4A}
fp_{1B}	fp_{2B}	fp_{3B}	fp_{3B}

a Cuckoo filter

113

113

Detecting Credential Stuffing ::

Our PMT :: Construction

Requester
(Element: ***"bigbang"***, *pk, sk*)

Responder
(Set: ***S***)

Set ***S*** = {"***bigbang***", "***domingo***", … }

"bigbang" $\xrightarrow{h_1(),\ h_2()}$

0	***1***
0	0
1	0
0	0

fp()

– f = – fp(*"bigbang"*)

S $\xrightarrow[\text{fp()}]{h_1(),\ h_2()}$

fp_{1A}	fp_{2A}	fp_{3A}	fp_{4A}
fp_{1B}	fp_{2B}	fp_{3B}	fp_{3B}

a Cuckoo filter

Encrypted under an additively homomorphic encryption scheme with public key *pk*.

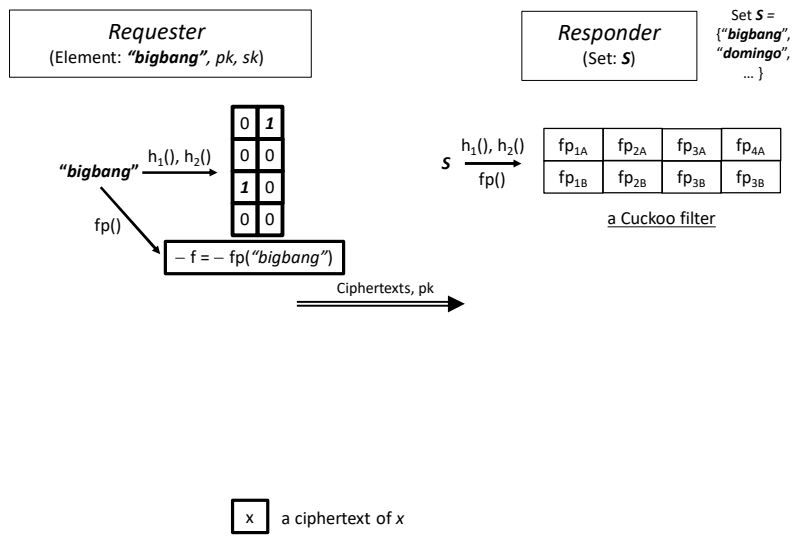
[x] a ciphertext of *x*

114

114

Detecting Credential Stuffing ::

Our PMT :: Construction



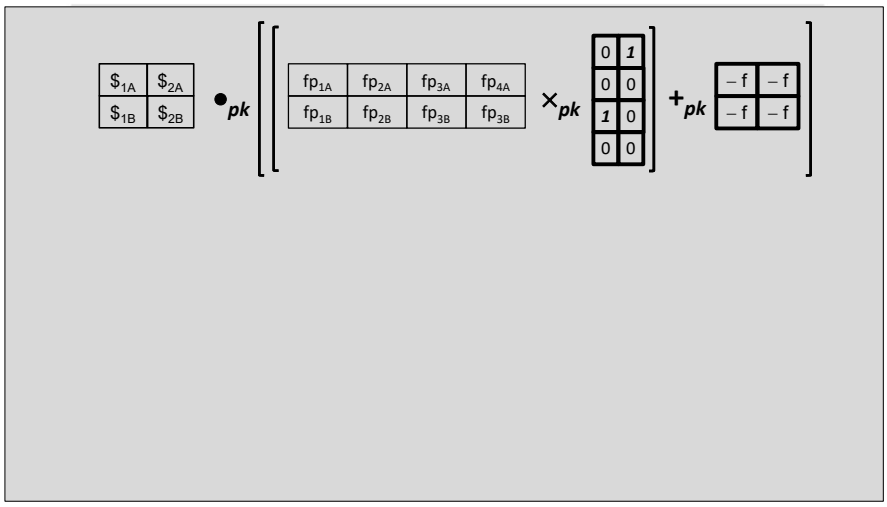
x a ciphertext of *x*

115

115

Detecting Credential Stuffing ::

Our PMT :: Construction



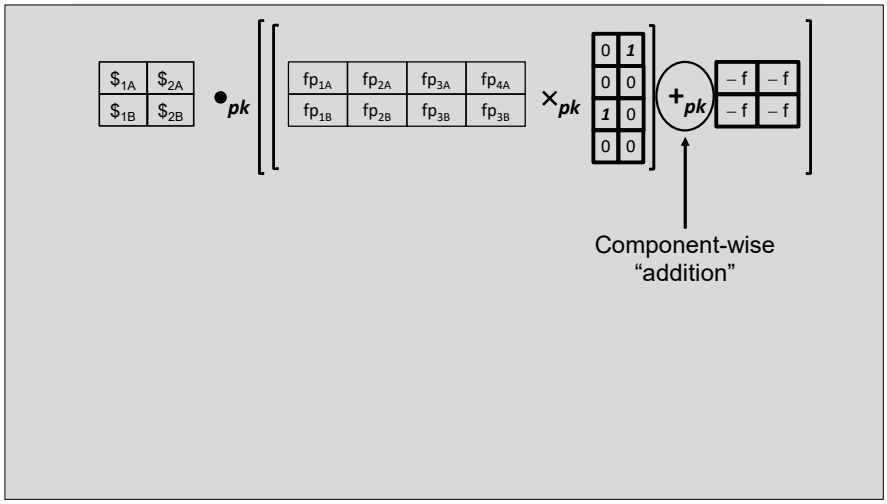
x a ciphertext of *x*

116

116

Detecting Credential Stuffing ::

Our PMT :: Construction



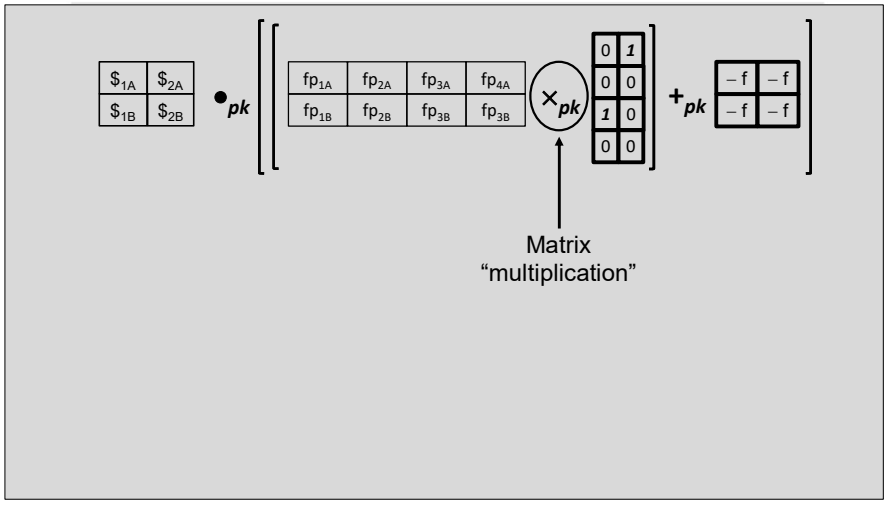
x a ciphertext of *x*

117

117

Detecting Credential Stuffing ::

Our PMT :: Construction



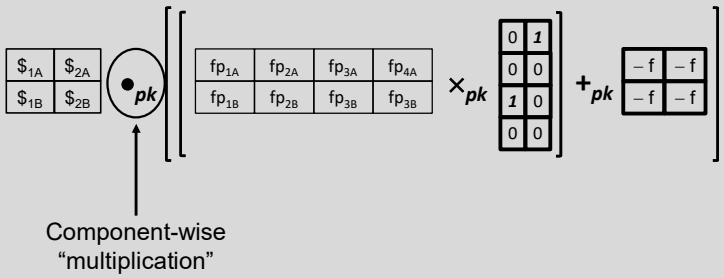
x a ciphertext of *x*

118

118

Detecting Credential Stuffing ::

Our PMT :: Construction



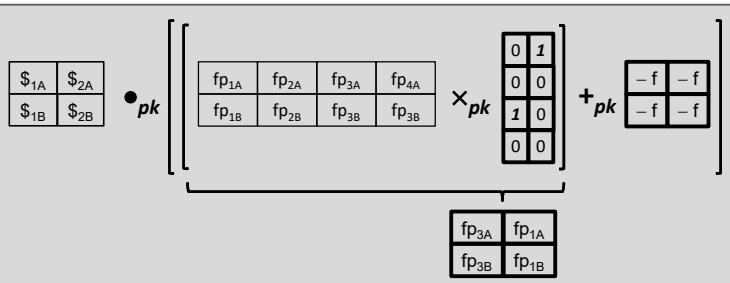
x a ciphertext of *x*

119

119

Detecting Credential Stuffing ::

Our PMT :: Construction



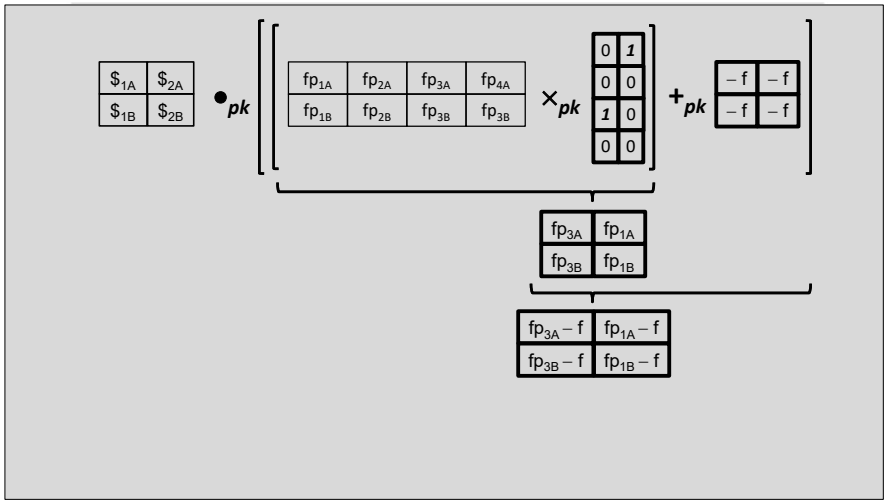
x a ciphertext of *x*

120

120

Detecting Credential Stuffing ::

Our PMT :: Construction



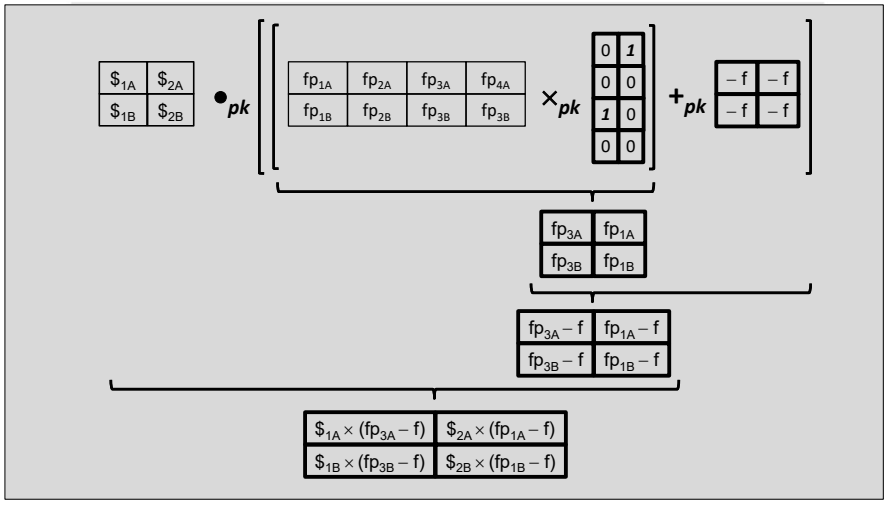
x a ciphertext of *x*

121

121

Detecting Credential Stuffing ::

Our PMT :: Construction



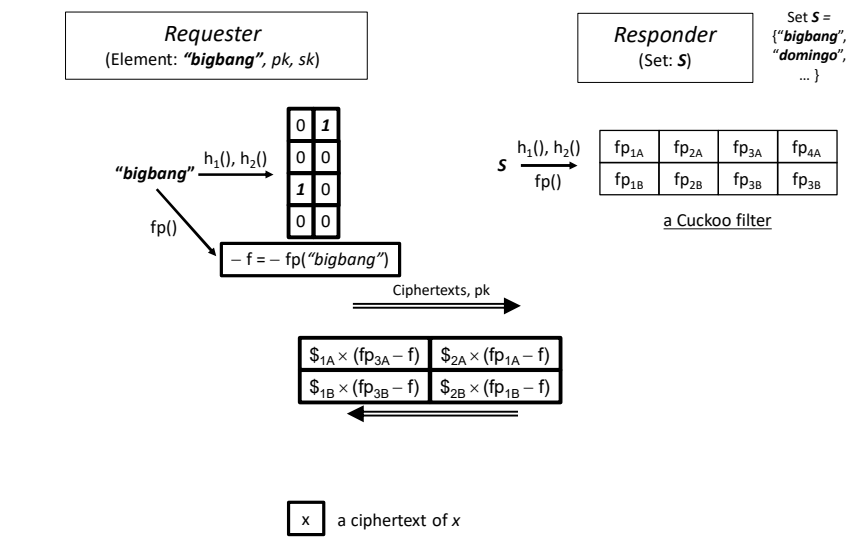
x a ciphertext of *x*

122

122

Detecting Credential Stuffing ::

Our PMT :: Construction

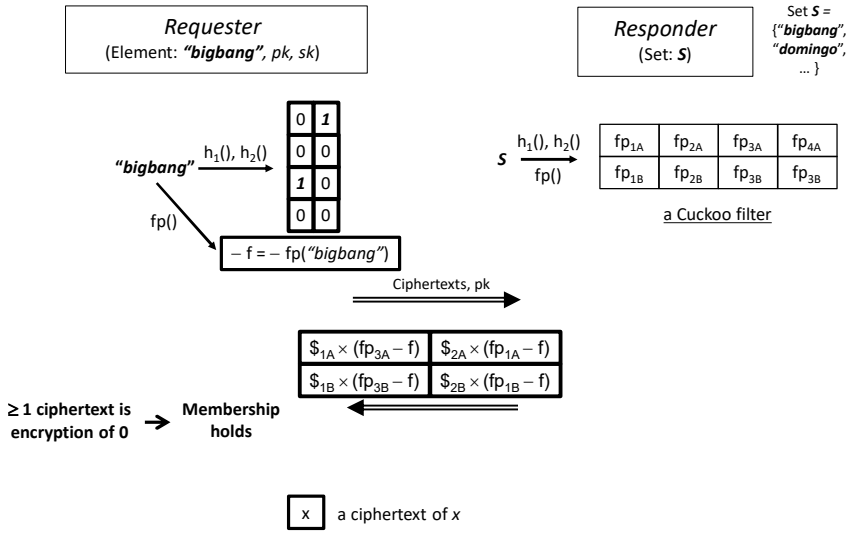


123

123

Detecting Credential Stuffing ::

Our PMT :: Construction



124

124

Detecting Credential Stuffing ::

Our PMT :: Properties

- **Rounds: One round of interaction, due to the integration of Cuckoo filters and partially homomorphic encryption**

125

125

Detecting Credential Stuffing ::

Our PMT :: Properties

- Rounds: One round of interaction, due to the integration of Cuckoo filters and partially homomorphic encryption
- **Response size: constant number of ciphertexts, due to the adoption of Cuckoo filters**

126

126

Detecting Credential Stuffing ::

Our PMT :: Properties

- Rounds: One round of interaction, due to the adoption of Cuckoo filters and partially homomorphic encryption schemes
- Response size: constant number of ciphertexts, due to the adoption of Cuckoo filters
- **Requester privacy: Information leakage to a malicious responder is negligible if the underlying encryption scheme is CPA secure**

127

127

Detecting Credential Stuffing ::

Our PMT :: Properties

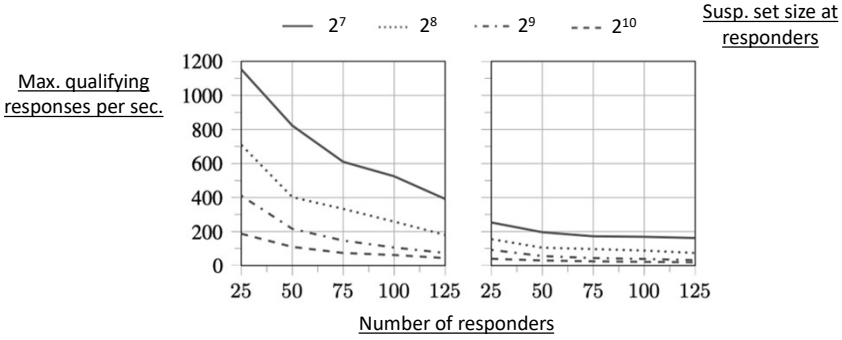
- Rounds: One round of interaction, due to the adoption of Cuckoo filters and partially homomorphic encryption schemes
- Response size: constant number of ciphertexts, due to the adoption of Cuckoo filters
- Requester privacy: Information leakage to a malicious responder is negligible if the underlying encryption scheme is CPA secure
- **Responder privacy: Information leakage to a malicious requester is not significantly more than the answer to the PMT query**

128

128

Detecting Credential Stuffing ::

Scalability



Trusted directory for *login privacy* (Qualifying response: <= 5s)

Untrusted directory for *login privacy* (Qualifying response: <= 8s)

Respose time measured at the requester

129

129

Detecting Credential Stuffing ::

Scalability

	Credential-stuffing login attempts per day	Proportion that succeed	Proportion of all login attempts
Airline	1.4 Million	1.00%	60%
Hotel	4.3 Million	1.00%	44%
Retail	131.5 Million	0.50%	91%
Consumer banking	232.2 Million	0.05%	58%

Table: Credential stuffing estimates for four major U.S. industries[6]

Total number of PMT queries per second:

- *If ADS false & true detection rates are 0.30 & 0.95 (against phishing attackers):* ***660***
- *If ADS false & true detection rates are 0.10 & 0.99 (against researching attackers):* ***227***

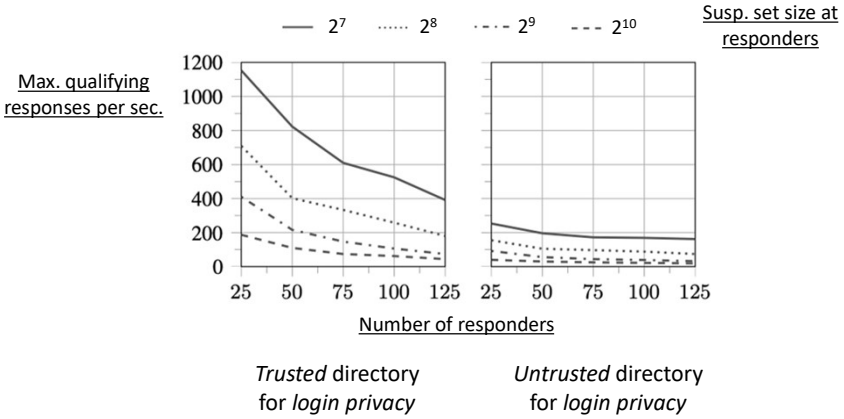
[6] Shape Security, "2018 Credential spill report'

130

130

Detecting Credential Stuffing ::

Scalability



Trusted directory for *login privacy*

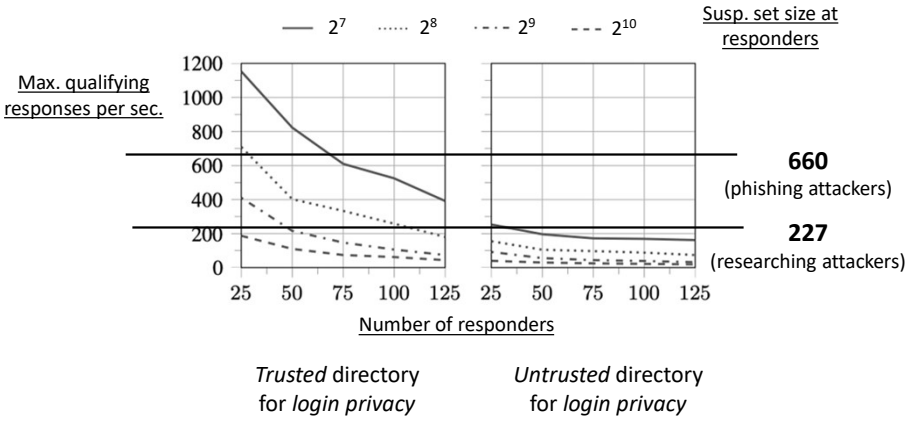
Untrusted directory for *login privacy*

131

131

Detecting Credential Stuffing ::

Scalability



Trusted directory for *login privacy*

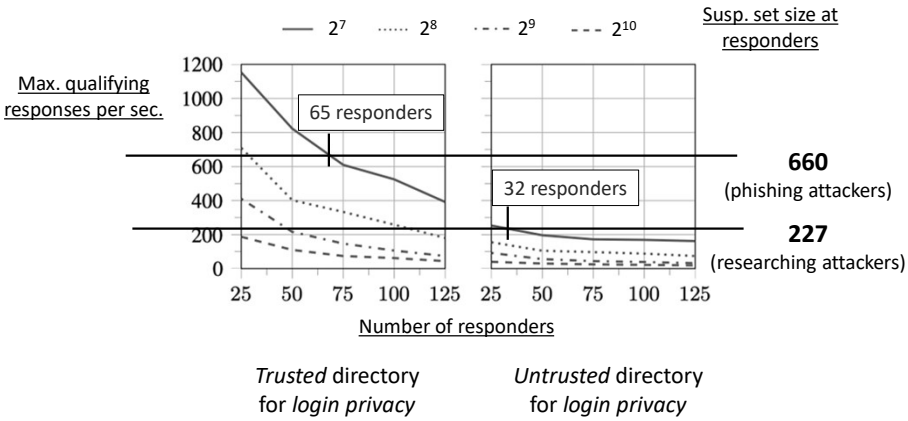
Untrusted directory for *login privacy*

132

132

Detecting Credential Stuffing ::

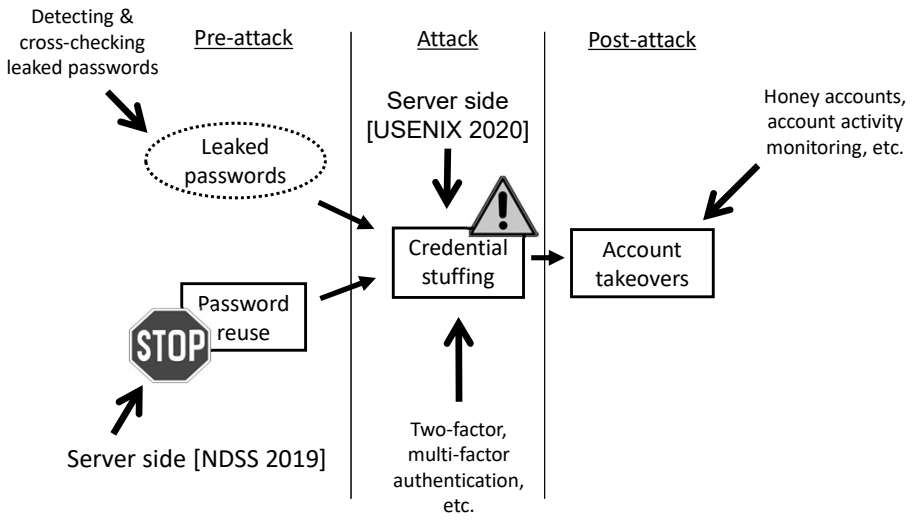
Scalability



133

133

Our Work



134

134

Parting Thoughts

- Coordination across websites for improving user security is (IMO) a rich opportunity
- Where and how can it be done *securely*? *Privately*? In a way that gains *acceptance* from users?
- How much user inconvenience is too much?

135

135

Parting Thoughts

- Coordination across websites for improving user security is (IMO) a rich opportunity
- Where and how can it be done *securely*? *Privately*? In a way that gains *acceptance* from users?
- How much user inconvenience is too much?

Thank you!

136

136