An Empirical Study on Mobile Payment Credential Leaks and Their Exploits

Shangcheng Shi, Xianbo Wang, Kyle Zeng,

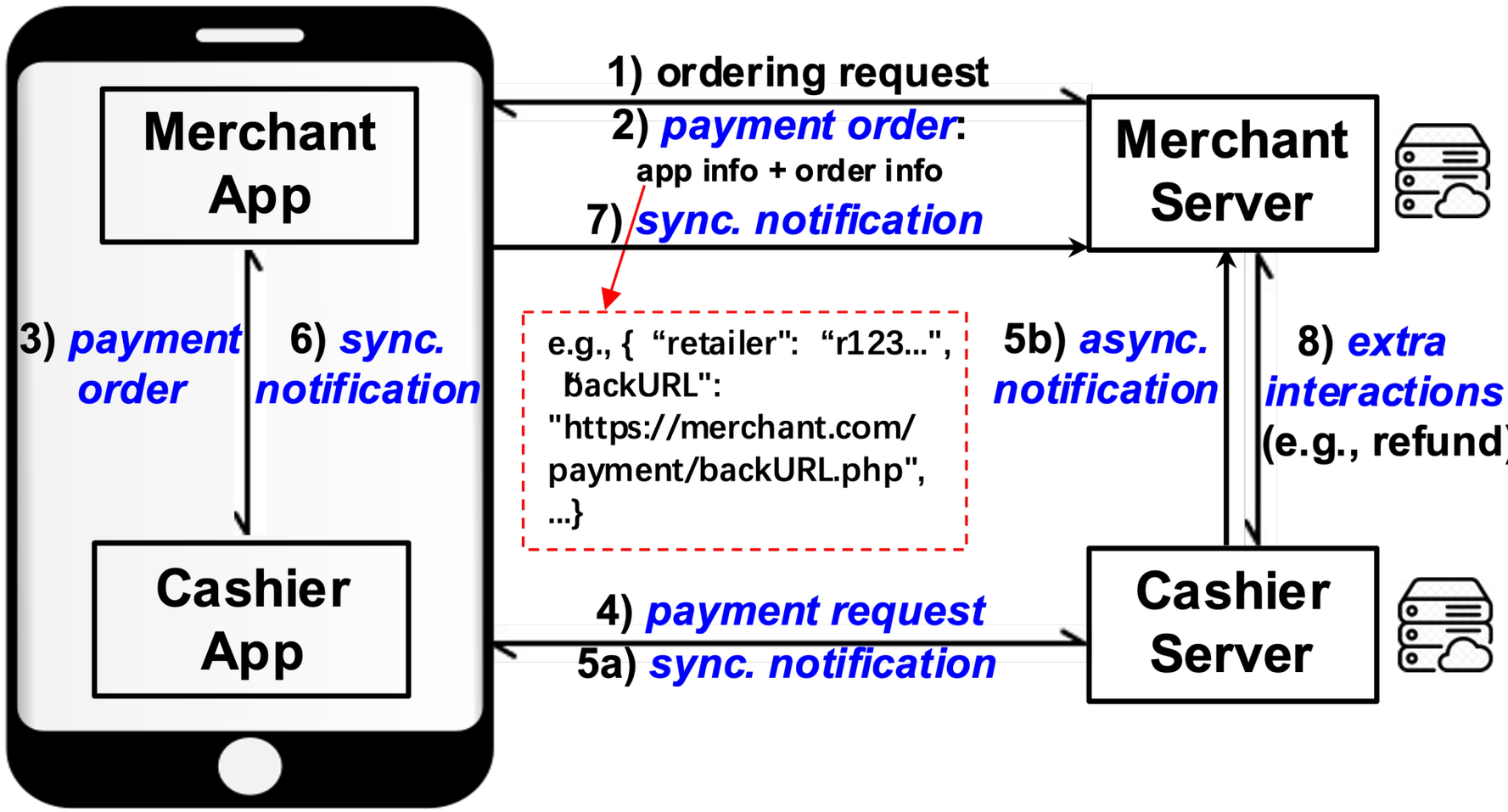
Ronghai Yang and Wing Cheong Lau

The Chinese University of Hong Kong

Outline

- Introduction to Mobile Payment Service and Credentials
- Leaking Sources of Payment Credentials
- Exploiting Leaked Payment Credentials
- Automated Mining for Payment Credentials
- Empirical Testing with PayKeyMiner

Third-Party Mobile Payment Service



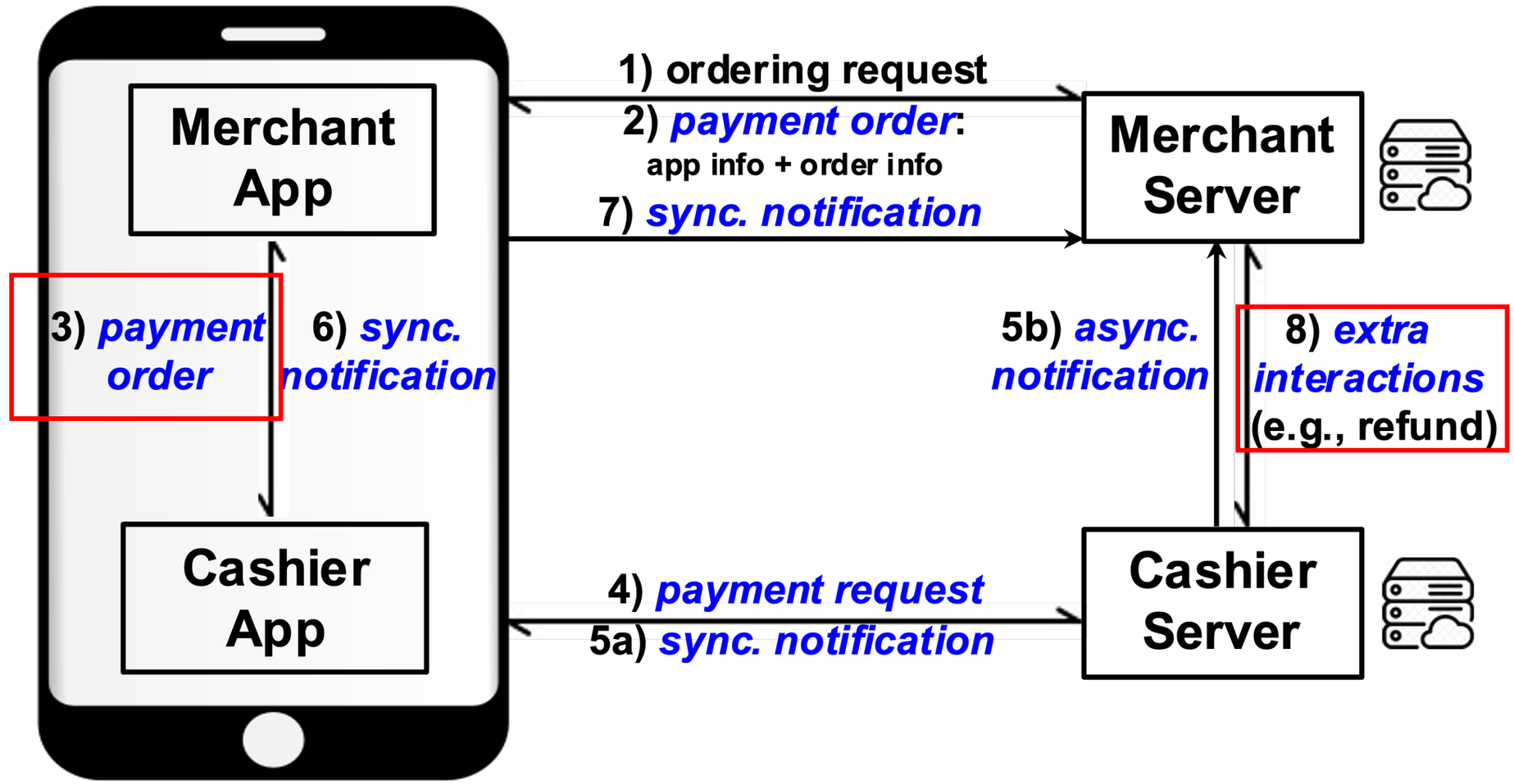
- The user can pay the Merchant App through the Cashier.
- The messages in italic are secured cryptographically.

Payment Credentials: Payment Key

Cashier	Payment Credential	Usage	Assigned by the Cashier?	Shared Cashier's Public Key
Cashier1	Secret Key	HMAC	✓	N/A
	RSA (Private) Key	Digital Signature	✗	✓
	RSA' (Private) Key	Digital Signature	✗	✗
Cashier2	Secret Key	HMAC	✗	N/A
Cashier3	Secret Key	HMAC	✓	N/A
	PFX Certificate	Digital Signature	✓	✓
Cashier4	Secret Key	HMAC	✓	N/A

- The Cashiers define payment keys for the HMAC or digital signature.
- The setting of these credentials differs among the Cashiers.

Payment Credentials: Other Credentials



- Android Signing Key (in *Cashier2* & *Cashier4*)
- SSL Client Certificate (in *Cashier2*)

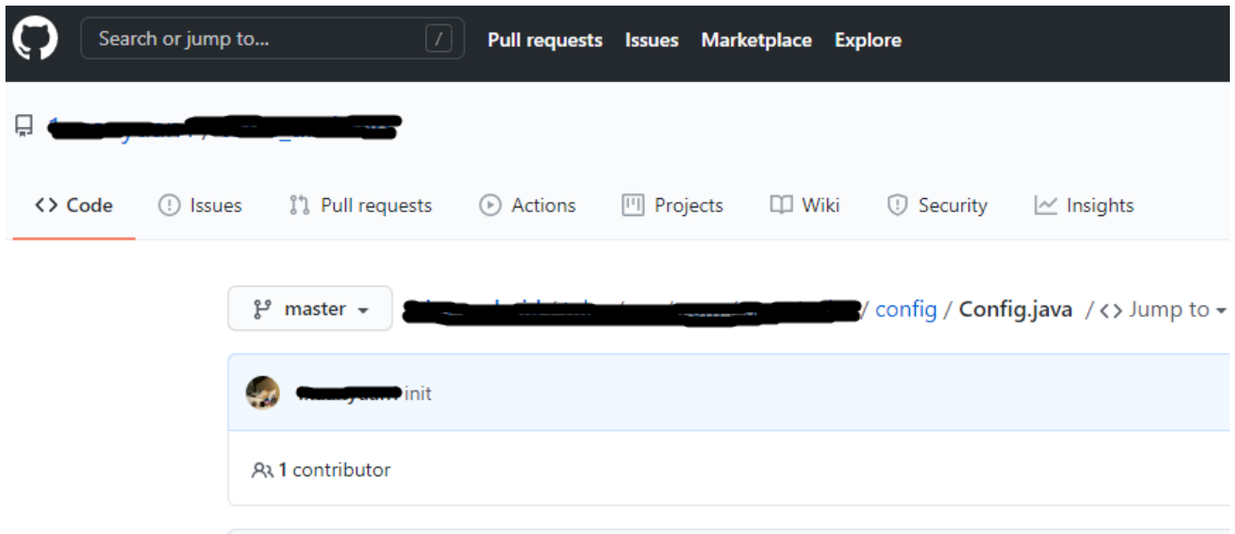
Outline

- Introduction to Mobile Payment Service and Credentials
- Leaking Sources of Payment Credentials
- Exploiting Leaked Payment Credentials
- Automated Mining for Payment Credentials
- Empirical Testing with PayKeyMiner

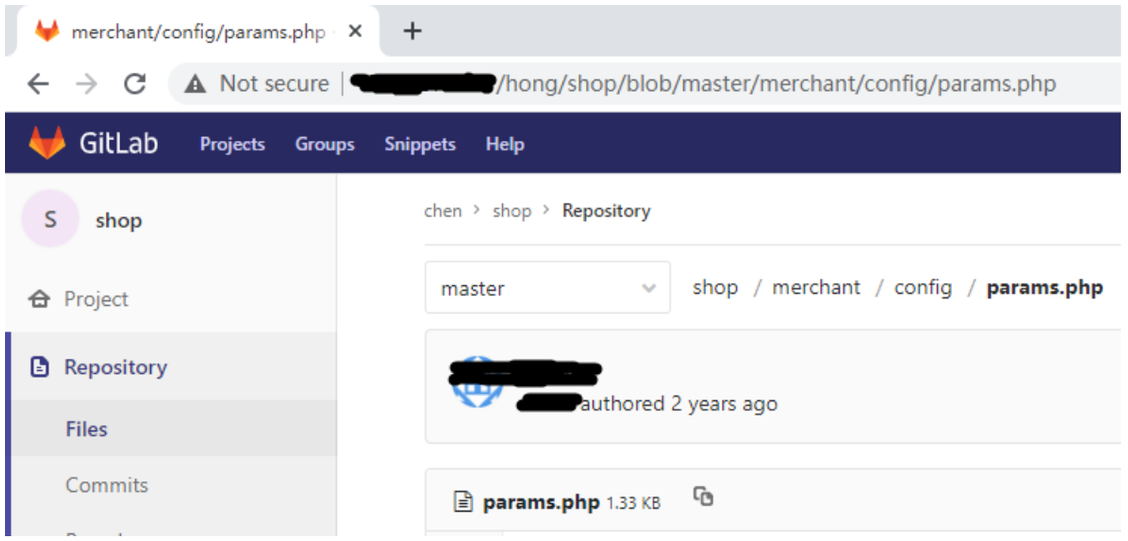
Leaking Sources of Payment Credentials

- Public Git Repositories

(1) GitHub



(2) GitLab

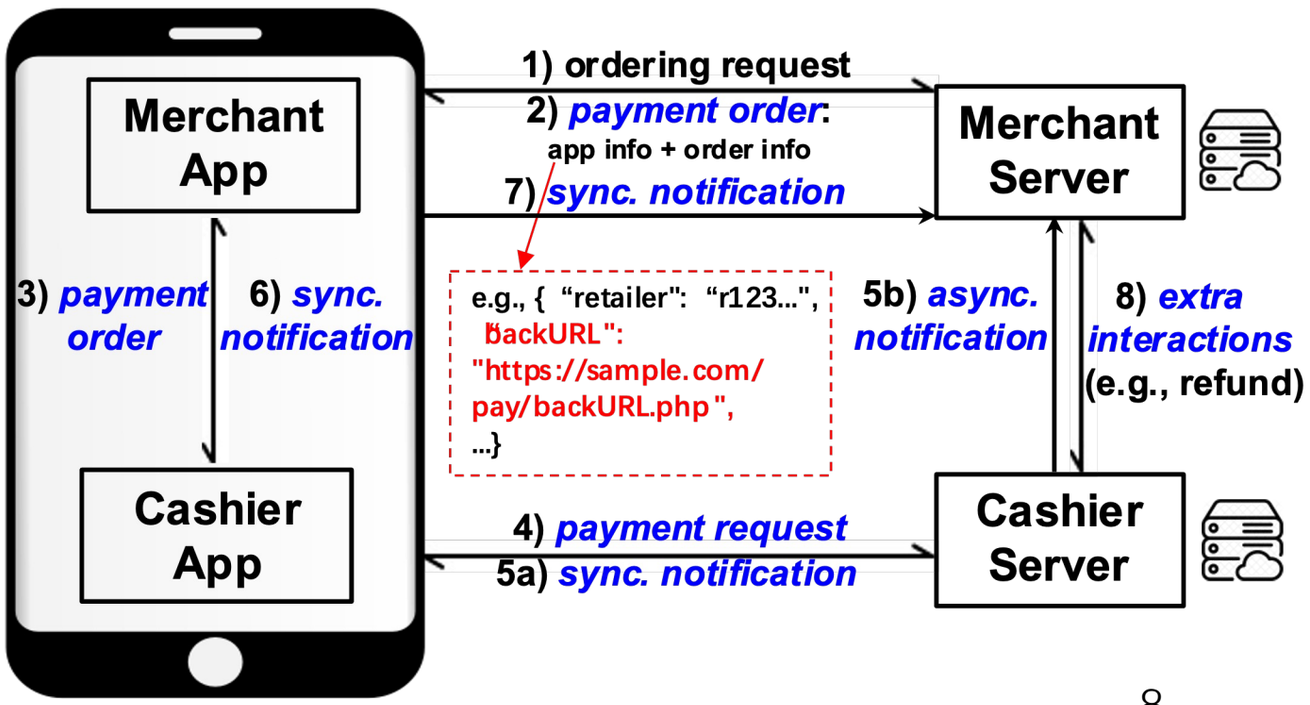


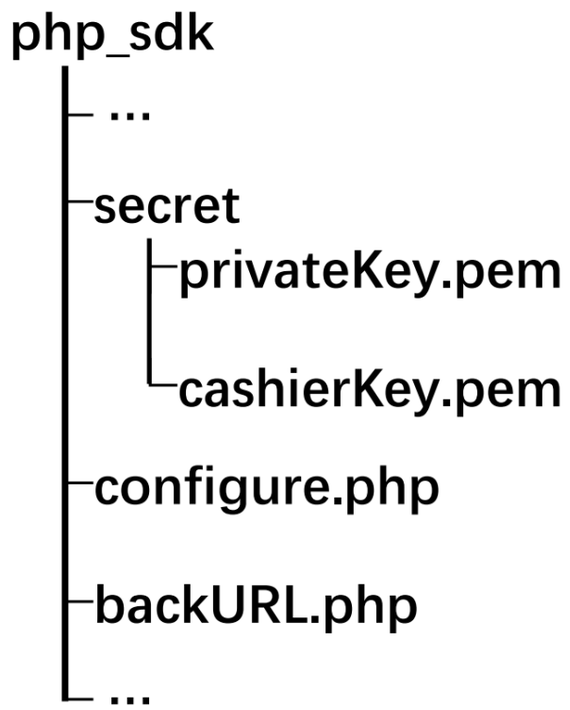
- Mobile Apps (e.g., Android APKs)

Leaking Sources of Payment Credentials

- Merchant Servers
  - Caused by (1) flawed backend SDKs (2) lack of access control on credential files
  - The attacker can infer the endpoint of the credential file according to *backURL,* e.g

https://sample.com/pay/backURL.php => https://sample.com/pay/secret/privateKey.p



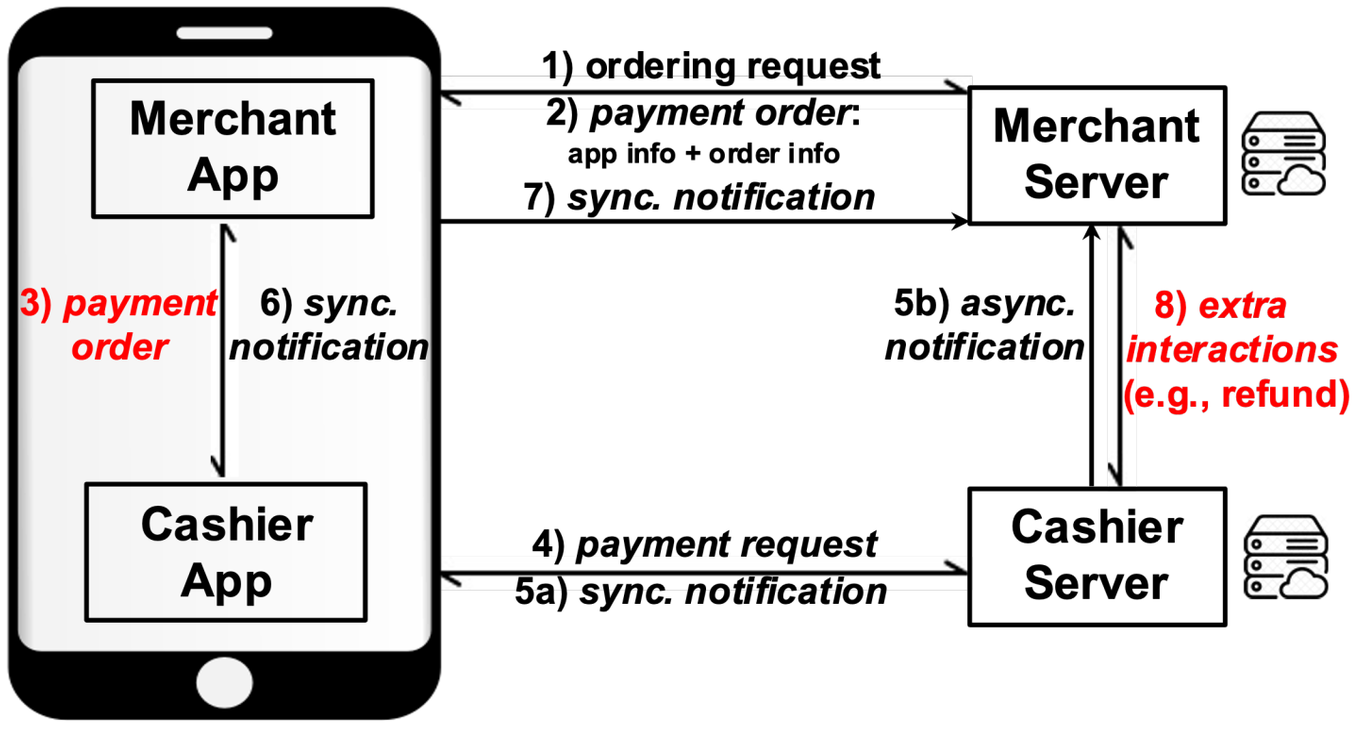


Outline

- Introduction to Mobile Payment Service and Credentials
- Leaking Sources of Payment Credentials
- Exploiting Leaked Payment Credentials
- Automated Mining for Payment Credentials
- Empirical Testing with PayKeyMiner

Exploiting Leaked Payment Credentials

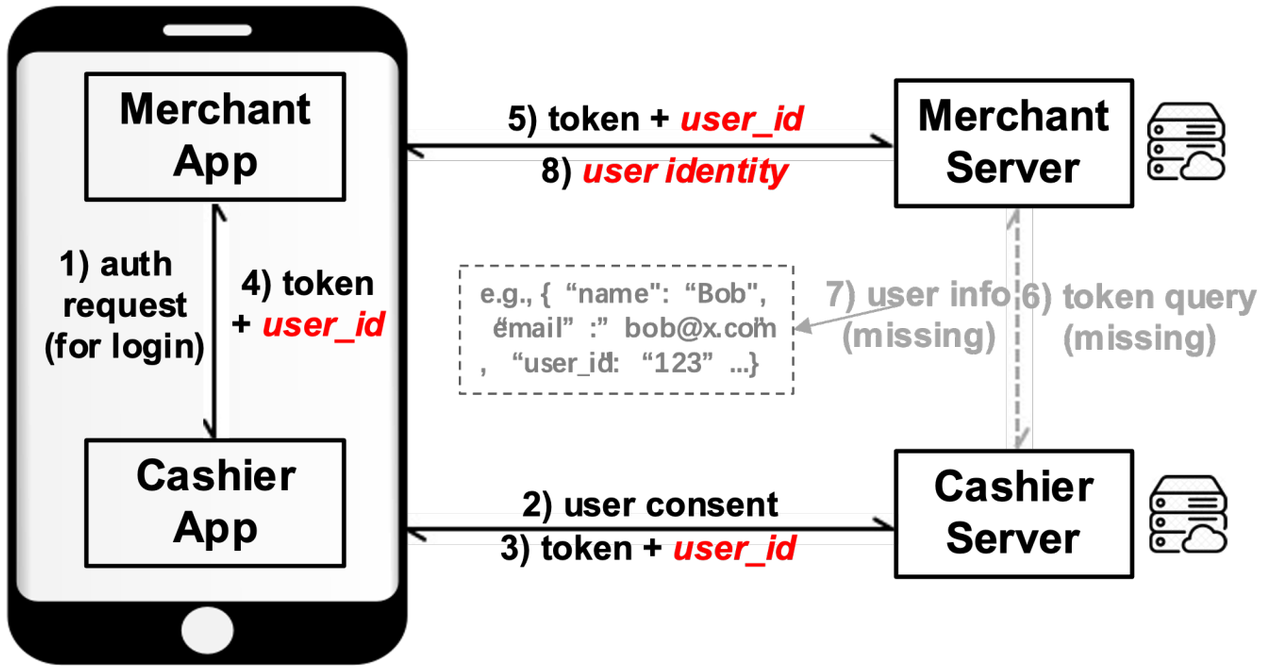
- Merchant Impersonation Exploit:
  - (1) Downloading Transaction Record (2) Refund (3) Money Transfer



- Android Package Signature Forgery:
  - Overall, 400+ valid Android signing keys have been detected.

Exploiting Leaked Payment Credentials

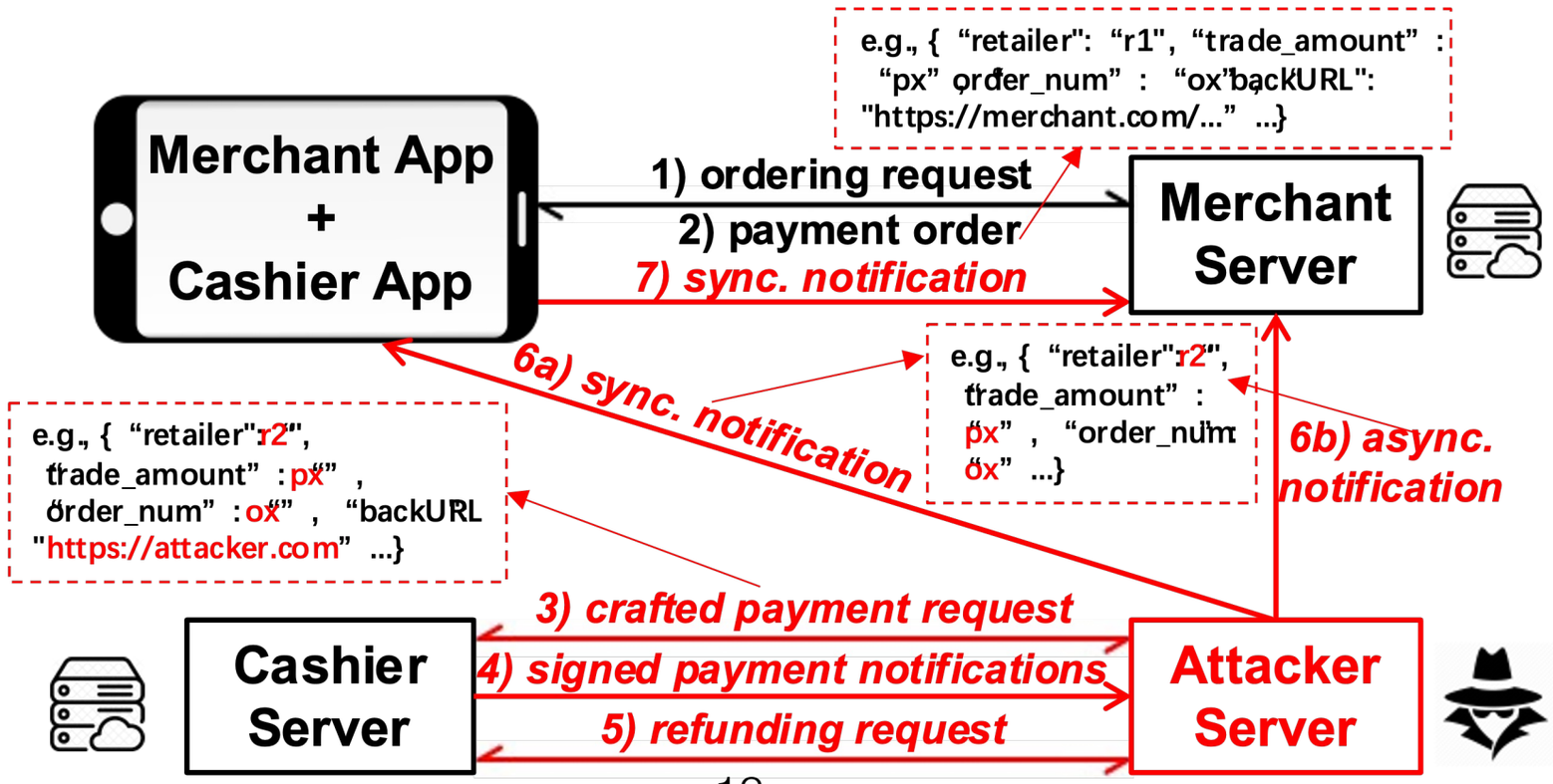
- Backward SSO Attack:
  - Two Cashiers offer SSO service but fail to isolate their services, e.g., shared *user_id*s.
  - The attacker may hijack the victim's Merchant account with Profile Exploit [1].
  - Reusage of payment keys as the SSO credentials



[1] R. Yang, W. C. Lau and S. Shi, "Breaking and Fixing Mobile App Authentication with OAuth2.0-based Protocols" in ACNS, 2017

Exploiting Leaked Payment Credentials

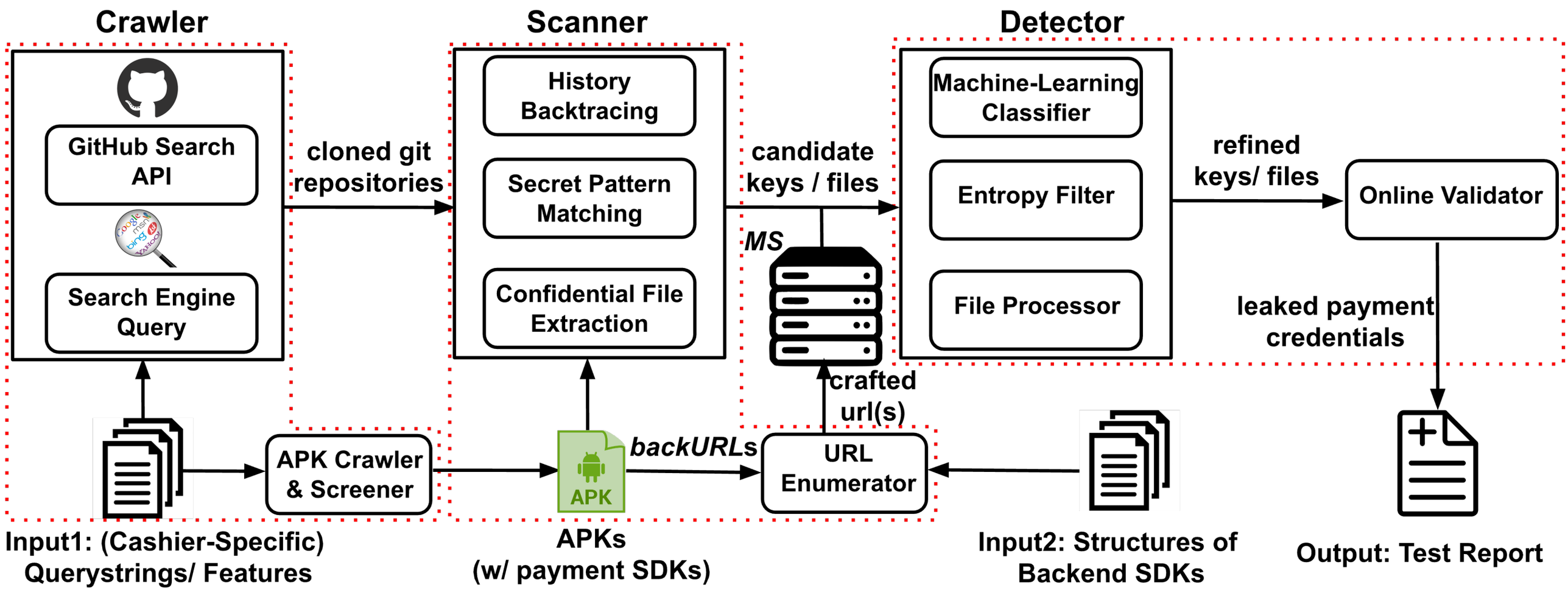
- Cross-App Payment Notification Forgery:
  - When using the digital signature, the public key of the Cashier tends to be shared.
  - Some Merchant Server overlooks the app identifier in the payment notifications.
  - The attacker may forge payment notifications to cheat another Merchant App.



Outline

- Introduction to Mobile Payment Service and Credentials
- Leaking Sources of Payment Credentials
- Exploiting Leaked Payment Credentials
- Automated Mining for Payment Credentials
- Empirical Testing with PayKeyMiner

PayKeyMiner



- We develop an automated tool to enable large-scale mining for the payment credentials leaked in the wild.

Outline

- Introduction to Mobile Payment Service and Credentials
- Leaking Sources of Payment Credentials
- Exploiting Leaked Payment Credentials
- Automated Mining for Payment Credentials
- Empirical Testing with PayKeyMiner

Empirical Testing Result

Cashier	*Cashier1*			*Cashier2*			*Cashier3*		*Cashier4*	
Source \ Credential	Secret Key	RSA Key	RSA' Key	Secret Key	Client Cert	Android Key	Secret Key	PFX Cert	Secret Key	Android Key
GitHub Repo	900	1518	1737	6651	3131	491	0	188	25	1
GitLab Repo	9	20	20	57	31	1	0	1	0	0
Android APK	75	1950	354	2567	3	0	2	0	10	0
Merchant Server	N/A	44	0	N/A	11	N/A	0	2	0	N/A
Overall	975	3332	2085	9093	3170	492	2	189	34	1

- PayKeyMiner has detected roughly 20,000 unique payment credentials leaked from different sources.

Empirical Testing Result

- Public Git Repositories:
  - 7.8% of the credentials are from old git commits.
  - Over 700 payment credentials are related to iOS apps.
  - Most public GitLab repositories are owned by some outsourcing companies.
- Android APKs:
  - Overall, 4,961 unique payment credentials have been detected.
  - 31.9% of these credentials are from the old app versions only.
- Merchant Servers:
  - We use HTTP HEAD to probe these exposed credential files without downloading them.
  - 7.1% percent of the tested servers fail to protect their credentials.

Longitudinal Study

Cashier	Cashier1		Cashier2	
Fixing Methods	3 months later	12 months later	3 months later	12 months later
#Updating the Leaked Key	2 (0.3%)	255 (35.5%)	337 (9.2%)	443 (12.1%)
#Hiding the GitHub Repo	127 (17.7%)	146 (20.3%)	377 (10.3%)	651 (17.8%)
#Deleting Git Commits	117 (16.3%)	65 (9.1%)	218 (6.0%)	198 (5.4%)
#Pushing New Git Commits	8 (1.1%)	3 (0.4%)	29 (0.8%)	24 (0.7%)
#No Response	464 (64.6%)	249 (34.7%)	2701 (73.8%)	2346 (64.1%)
#Detected Key (#Unique Key)	718 (624)		3662 (2728)	

- We reported 3,000+ payment keys to the Cashiers after our initial testing.
- We regularly monitor these submitted keys to study the responses from the Merchants.
- Around 60% of the leaking Merchants have not made any response.

Suggested Fixes

- We give the following suggestions to mitigate the payment credential leaks:

- (1) The Cashiers should alarm their Merchants about the serious consequences of payment credential leaks.

- (2) The Cashiers should review their services and timely fix the insecure implementations, including the vulnerable backend SDKs and shared *user_id*s.

- (3) The Cashiers should proactively detect and revoke the leaked credentials

- (4) The Merchants had better periodically update their payment credentials.

Thanks!
Q&A