Review “ContextIoT”

Name: Wenhui Zhang

Paper Name：

Jia, Y. J., Chen, Q. A., Wang, S., Rahmati, A., Fernandes, E., Mao, Z. M., ... & Unviersity, S. J. (2017). ContexIoT: Towards providing contextual integrity to appified IoT platforms. In *Proceedings of The Network and Distributed System Security Symposium* (Vol. 2017).

Contribution:

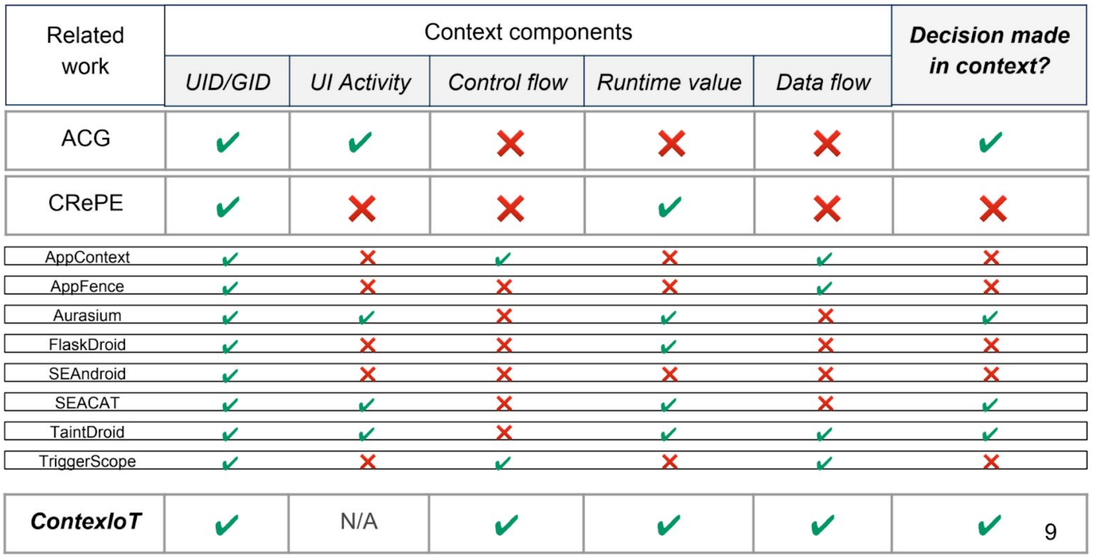
This paper deploys system that provides contextual permission prompts in SmartThings apps, ContexIoT. ContexIoT is a context-based permission system for appified IoT platforms that provides contextual integrity by supporting fine-grained context identification for sensitive actions, and runtime prompts with rich context information to help users perform effective access control.

Motivation:

Design flaws in current IoT platform permission models have been reported recently, exposing users to significant harm such as break-ins and theft. Thus, a new access control model is needed for both current and future IoT platforms.

Related works:

Context concept various as following:



Related work	Context components					*Decision made in context?*
	UID/GID	*UI Activity*	*Control flow*	*Runtime value*	*Data flow*	
ACG	✓	✓	✗	✗	✗	✓
CRePE	✓	✗	✗	✓	✗	✗
AppContext	✓	✗	✓	✗	✓	✗
AppFence	✓	✗	✗	✗	✓	✗
Aurasium	✓	✓	✗	✓	✗	✓
FlaskDroid	✓	✗	✗	✓	✗	✗
SEAndroid	✓	✗	✗	✗	✗	✗
SEACAT	✓	✓	✗	✓	✗	✓
TaintDroid	✓	✓	✗	✓	✓	✓
TriggerScope	✓	✗	✓	✗	✓	✗
ContexIoT	✓	N/A	✓	✓	✓	✓ 9

Methodology:

Context definition in ContexIoT is at the inter-procedure control and data flow levels.

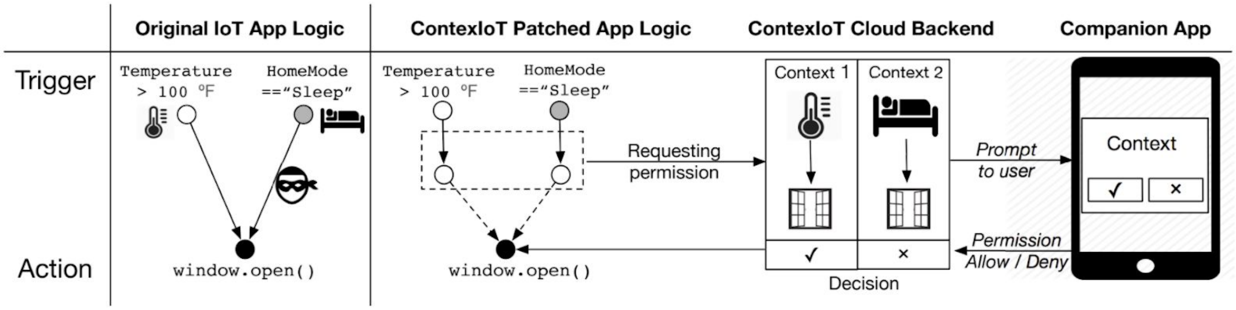


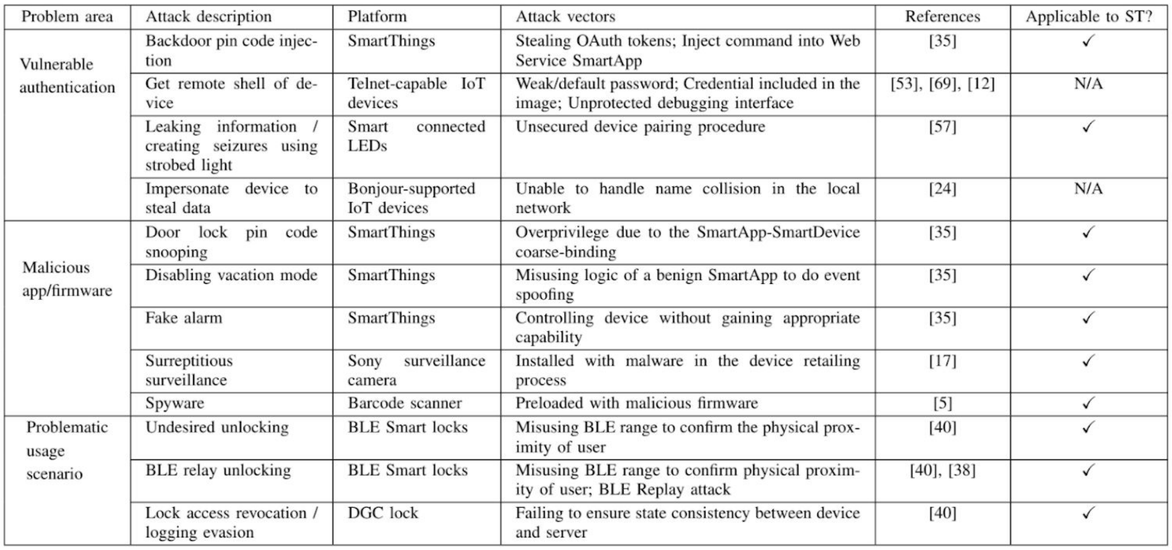
Fig. 2: ContexIoT overview with a concrete example showing our context-based access control

Results:

Evaluation is done on data set of Attacks Migrated From Mobile Platform (https://sites.google.com/site/iotcontextualintegrity/attack/existing-iot-attacks).

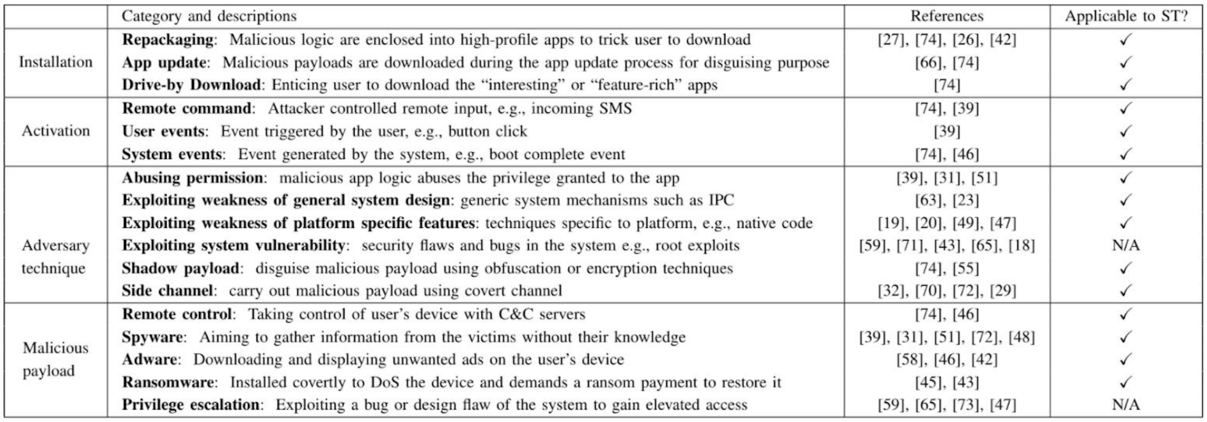
Take away:

TABLE I: A taxonomy of reported IoT attacks and their applicability to the SmartThings platform



Problem area	Attack description	Platform	Attack vectors	References	Applicable to ST?
Vulnerable authentication	Backdoor pin code injection	SmartThings	Stealing OAuth tokens; Inject command into Web Service SmartApp	[35]	✓
	Get remote shell of device	Telnet-capable IoT devices	Weak/default password; Credential included in the image; Unprotected debugging interface	[53], [69], [12]	N/A
	Leaking information / creating seizures using strobed light	Smart connected LEDs	Unsecured device pairing procedure	[57]	✓
	Impersonate device to steal data	Bonjour-supported IoT devices	Unable to handle name collision in the local network	[24]	N/A
Malicious app/firmware	Door lock pin code snooping	SmartThings	Overprivilege due to the SmartApp-SmartDevice coarse-binding	[35]	✓
	Disabling vacation mode	SmartThings	Misusing logic of a benign SmartApp to do event spoofing	[35]	✓
	Fake alarm	SmartThings	Controlling device without gaining appropriate capability	[35]	✓
	Surreptitious surveillance	Sony surveillance camera	Installed with malware in the device retailing process	[17]	✓
	Spyware	Barcode scanner	Preloaded with malicious firmware	[5]	✓
Problematic usage scenario	Undesired unlocking	BLE Smart locks	Misusing BLE range to confirm the physical proximity of user	[40]	✓
	BLE relay unlocking	BLE Smart locks	Misusing BLE range to confirm physical proximity of user; BLE Replay attack	[40], [38]	✓
	Lock access revocation / logging evasion	DGC lock	Failing to ensure state consistency between device and server	[40]	✓

TABLE II: A taxonomy of smartphone malware classes and their applicability to the SmartThings platform



	Category and descriptions	References	Applicable to ST?
Installation	**Repackaging**: Malicious logic are enclosed into high-profile apps to trick user to download	[27], [74], [26], [42]	✓
	App update: Malicious payloads are downloaded during the app update process for disguising purpose	[66], [74]	✓
	Drive-by Download: Enticing user to download the “interesting” or “feature-rich” apps	[74]	✓
Activation	**Remote command**: Attacker controlled remote input, e.g., incoming SMS	[74], [39]	✓
	User events: Event triggered by the user, e.g., button click	[39]	✓
	System events: Event generated by the system, e.g., boot complete event	[74], [46]	✓
Adversary technique	**Abusing permission**: malicious app logic abuses the privilege granted to the app	[39], [31], [51]	✓
	Exploiting weakness of general system design: generic system mechanisms such as IPC	[63], [23]	✓
	Exploiting weakness of platform specific features: techniques specific to platform, e.g., native code	[19], [20], [49], [47]	✓
	Exploiting system vulnerability: security flaws and bugs in the system e.g., root exploits	[59], [71], [43], [65], [18]	N/A
	Shadow payload: disguise malicious payload using obfuscation or encryption techniques	[74], [55]	✓
	Side channel: carry out malicious payload using covert channel	[32], [70], [72], [29]	✓
Malicious payload	**Remote control**: Taking control of user’s device with C&C servers	[74], [46]	✓
	Spyware: Aiming to gather information from the victims without their knowledge	[39], [31], [51], [72], [48]	✓
	Adware: Downloading and displaying unwanted ads on the user’s device	[58], [46], [42]	✓
	Ransomware: Installed covertly to DoS the device and demands a ransom payment to restore it	[45], [43]	✓
	Privilege escalation: Exploiting a bug or design flaw of the system to gain elevated access	[59], [65], [73], [47]	N/A