Research Project Exploration

March 19, 2020

1. Extend bug patterns from one project to another

- A guess: A programmer tends to make similar mistakes cross projects
- If for bug finding:
    - How to define a bug pattern? bug type, code structure? (already studed)
    - How to define two bugs belong to same bug pattern. (can be improved, but not very meaningful)
    - Q: If have identified a bug pattern in a project, why not directly use the detection system to find bugs in other projects.
- If for investigating the commonness of the guess.
    - We do a large scale testing and show this is a very common fact
    - Specialize detection for each user according to his bug patterns when they write code

2.Hidden information on web(more than we can see)

- An observation: HTML/CSS scripts responded from a web server are processed locally in client-side browsers, which contain more information than a user requests.
- This is a feature of web to move some processing to client-side, and relieve the computation burden of web-server.
- Questions:
  - Security: Any privacy concern like explosing too much information?
  - System: Modify server to only respond necessary/viewable content to clients? (overhead)? (might not be doable)

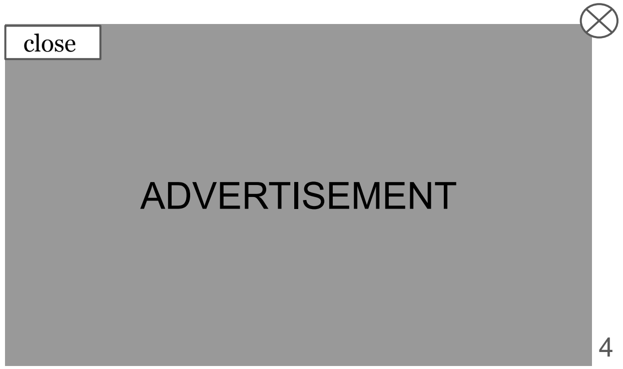
```
<style>
h1.hidden {
  display: none;
}
</style>
```

```
<!--  comments -->
```

3. UI deception on web

- Click the “x” on upper right corner might redirect to another website, while the real close is on the upper left corner.
- Questions:
  - Design UI deception based on use habits? (1. windows user might get used to click the upper right corner to close 2. the close button is more “attractive/obvious” than the text)
  - Whether users mistakenly click and how often? (User study)

- NDSS 2020: Deceptive Previews: A Study of the Link Preview Trustworthiness in Social Platforms
  - obtain benign-looking previews for malicious links.

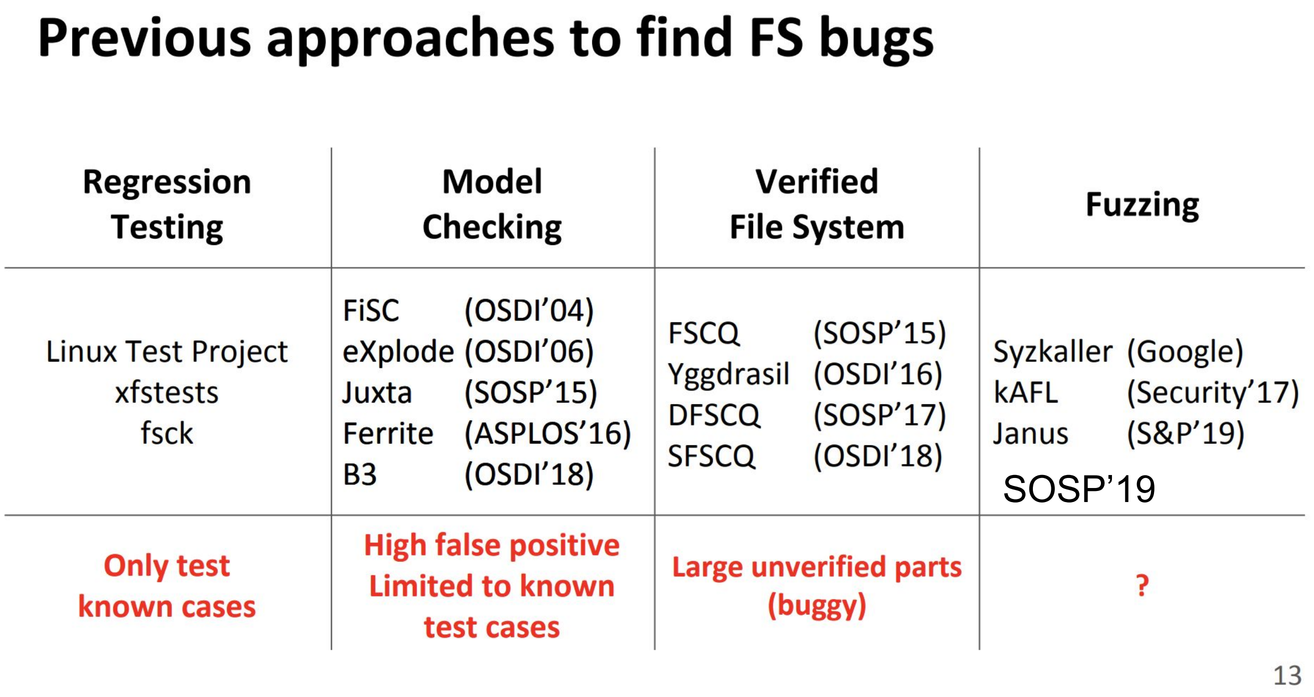


4. Automatic Customer Service

- Many phone/web apps have automatic customer service before manual customer service. This is to reduce human labor/money.
- Automatic customer service tries to understand the texts/voices of customers and feedbacks with several options/answers.
- Questions:
  - How accurate the techniques (NLP/SR) recognize the user inputs?
  - How users feel about it? Satisfied or not. (User study)
- Idea comes from a NDSS 2020 paper: Into the Deep Web: Understanding E-commerce Fraud from Autonomous Chat with Cybercriminals, which designs a system to chat with real-world e-commerce miscreants (e.g. QQ fraudsters)

5. OS bug detection.

- I didn't know there are so many works about file systems in recent years. It might not be good time for us to start to work on file systems now.
- Let's turn the direction to I/O systems like drivers. It might not be well studied. (need more exploration)



Previous approaches to find FS bugs

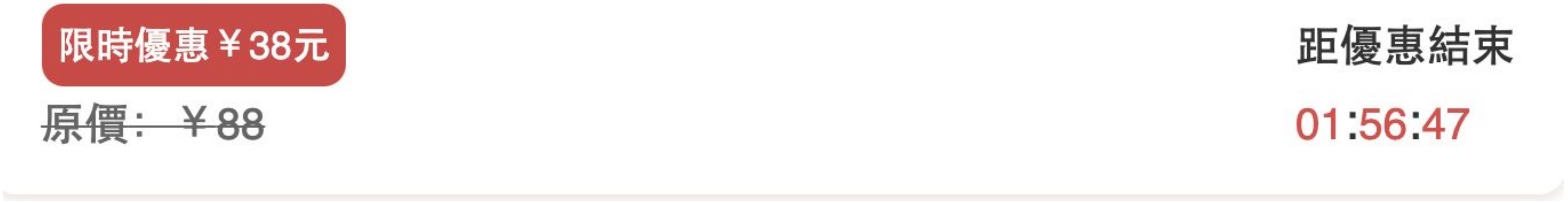
Regression Testing	Model Checking	Verified File System	Fuzzing
Linux Test Project xfstests fsck	FiSC (OSDI'04) eXplode (OSDI'06) Juxta (SOSP'15) Ferrite (ASPLOS'16) B3 (OSDI'18)	FSCQ (SOSP'15) Yggdrasil (OSDI'16) DFSCQ (SOSP'17) SFSCQ (OSDI'18)	Syzkaller (Google) kAFL (Security'17) Janus (S&P'19) SOSP'19
Only test known cases	**High false positive Limited to known test cases**	**Large unverified parts (buggy)**	**?**

13

6. Evaluate warnings in instant message Apps

- Some instant message apps generate warnings to notify unusual login (IP).
- Evaluate the technique and attack it!

7. Understanding this kind of fake discount


限時優惠￥38元

~~原價：￥88~~

距優惠結束

01:56:47

8. “Personalization”

- Different devices show different results
- Systematically investigate it in:
  - How different
  -

Several topics in NDSS/USENIX Security 2020

(D)DoS

- NDSS 2020: HotFuzz: Discovering Algorithmic Denial-of-Service Vulnerabilities Through Guided Micro-Fuzzing
  - Outperforms to slowfuzz: 1) no manual efforts 2) complicated seeds 3) Sanitizing inputs
- NDSS 2020: Poseidon: Mitigating Volumetric DDoS Attacks with Programmable Switches
- NDSS 2020: CDN Judo: Breaking the CDN DoS Protection with Itself

User study

- USENIX Security 2020: Understanding security mistakes developers make: Qualitative analysis from Build It, Break It, Fix It
    - investigate how and why programmers make security-relevant errors
- USENIX Security 2020: An Observational Investigation of Reverse Engineers' Processes
    - produce insights for improving interaction design for reverse engineering tools
- NDSS 2020: Are You Going to Answer That? Measuring User Responses to Anti-Robocall Application Indicators
    - how well anti-robocall application communicate risk with users

Fingerprinting

- USENIX Security 2020: Human Distinguishable Visual Key Fingerprints
- USENIX Security 2020: Zero-delay Lightweight Defenses against Website Fingerprinting
- NDSS 2020: Hold the Door! Fingerprinting Your Car Key to Prevent Keyless Entry Car Theft
- NDSS 2020: FlowPrint: Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic
- NDSS 2020: Carnus: Exploring the Privacy Threats of Browser Extension Fingerprinting

Fuzzing

- NDSS 2020: HYPER-CUBE: High-Dimensional Hypervisor Fuzzing
- NDSS 2020: HFL: Hybrid Fuzzing on the Linux Kernel
- NDSS 2020: HotFuzz: Discovering Algorithmic Denial-of-Service Vulnerabilities Through Guided Micro-Fuzzing
- NDSS 2020: Not All Coverage Measurements Are Equal: Fuzzing by Coverage Accounting for Input Prioritization
- USENIX Security 2020: Montage: A Neural Network Language Model-Guided JavaScript Engine Fuzzer
- USENIX Security 2020: FuzzGuard: Filtering out Unreachable Inputs in Directed Grey-box Fuzzing through Deep Learning
- USENIX Security 2020: GREYONE: Data Flow Sensitive Fuzzing

- Traditional and hard-core bug finding
  - Significantly improve existing work with new techniques
  - Extend/Define a new class of bug and show it is common
- Attacks
  - Craft an attack and show it’s severe