CACTI: CAPTCHA Avoidance via Client-side TEE Integration

Yoshimichi Nakatsuka\textsuperscript{1*}, Ercan Ozturk\textsuperscript{1*}, Andrew Paverd\textsuperscript{2}, Gene Tsudik\textsuperscript{1}

\textsuperscript{1}University of California, Irvine
\textsuperscript{2}Microsoft Research

*Primary Co-Authors
CAPTCHAs

**Completely Automated Public Turing test to tell Computers and Humans Apart**

- Tasks that are “easy” to solve by humans, yet “difficult” for machines

“Any program that has high success over a captcha can be used to solve an unsolved Artificial Intelligence (AI) problem”

Sample uses of CAPTCHAs

Protecting high-demand and/or limited items

- Event ticket sales

Protecting high-value events

- Signing up for a new account
- Voting in an online poll
Common CAPTCHA Types

reCAPTCHA (http://www.captcha.net/)

reCAPTCHA v2 (https://www.google.com/recaptcha/about/)

reCAPTCHA v3 (https://www.google.com/recaptcha/about/)
Downsides of CAPTCHAs

Difficult and time-consuming to solve
- Bursztein, et al. “How Good are Humans at Solving CAPTCHAs? A Large Scale Evaluation” IEEE Oakland 2010

Accessibility concerns
- Requires proper environment & device
- Blind or visually-impaired users?

Privacy concerns
- “Google’s new reCAPTCHA has a dark side” Fast Company 2019
- “Moving from reCAPTCHA to hCaptcha” Cloudflare 2020
Subverting CAPTCHAs

Modern machine learning can solve most types of CAPTCHAs
• Ye et al. “Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach” ACM CCS 2018
• Akrout et al. “Hacking Google reCAPTCHA v3 using Reinforcement Learning” arXiv 2019

CAPTCHA Forwarding attacks
• Bots forward CAPTCHAs to other sites to be solved by real users, e.g., discount coupon or porn sites

CAPTCHA farms
• Bots outsource CAPTCHA solving to human workers
• CAPTCHA solving services charge:
  • ~ $0.5 – $1 per 1000 CAPTCHAs
  • ~ $3 per 1000 reCAPTCHAs
CAPTCHAs are still widely used

According to trends.builtwith.com (as of Feb 2021)
• reCAPTCHA ≈ 6.37 million live sites
• reCAPTCHA v3 ≈ 1.59 million live sites

Claim: Despite their drawbacks, CAPTCHAs are still used to:
• increase attacker costs (in terms of time or money) and/or
• reduce the rate of malicious activities
Goals

- Minimize the number of CAPTCHAs shown to legitimate users without giving attackers a significant advantage

- Provide honest users a way to prove that they are not acting maliciously (for an appropriate definition of “maliciously”)
  - User: “I haven’t performed this action in the last $n$ hours”
  - Website: “Prove it”
  - User: “OK, here’s a proof”
CAPTCHA avoidance protocol

Instead of presenting a CAPTCHA, the following interaction takes place between the client (browser) and web server:

1) **Server** provides:
   - a rate threshold (i.e. # occurrences and a time period)
   - a timestamp for the current event

2) **Client** responds with proof that:
   - its rate for the specified action (within that time period) is below threshold, and
   - it has added the new timestamp to its database
CAPTCHA avoidance protocol

Client maintains both **per-website** lists of timestamps and a **global list**
- Websites can specify which list to use for the rate-proof
- List ownership enforced through cryptographic signatures

Clients that cannot (or do not want to) provide a rate-proof simply fall back to being presented with a CAPTCHA
Requirements and goals

Security
  • Clients cannot forge or modify rate-proofs

Privacy
  • A server (or a group thereof) cannot link rate-proofs to the clients that generated them, or link two rate-proofs to the same client

Deployability
  • Minimize user-perceived latency
  • Minimize data transfer between client and server
CAPTCHA avoidance using TEEs

If the threshold is satisfied:
1. Add timestamp $t$ to list
2. Generate rate-proof

“If there are no more than $k$ timestamps since $t_s$, store $t$ and provide a rate proof”

$\text{Signature using a group private key}$

$t$: New timestamp
$t_s$: Threshold starting time

$k$: Threshold count

$name$: Server name

GET example.com

Verify

CAPTCHA_PASS, example.com
Challenges & Solutions (1)

Challenge: limited amount of secure TEE memory
  • e.g. current SGX enclaves have 100 MB

Solution: store timestamps outside the enclave but ensure integrity using hash chains
  • only need integrity protection for the most recent hash

\[
H_0 = H(T_0) \quad \rightarrow \quad H_1 = H(H_0, T_1) \quad \rightarrow \quad \ldots \quad \rightarrow \quad H_{n+1} = H(H_n, T_{n+1})
\]
**Challenge**: limited number of monotonic counters
  • e.g. current consumer SGX CPUs allow 256 counters per enclave

**Solution**: use a Merkle hash tree over the heads of the hash chains
  • each leaf is the head of a website-specific hash-chain + list information
  • only need roll-back protection for the root of the tree
CACTI prototype implementation

Host Application
- SQLite
- Intel SGX Enclave

Browser
- CACTI Extension
- Background Script
- Content Script
- resource.html

<div id=…>
## Security evaluation

<table>
<thead>
<tr>
<th>Adversarial client</th>
<th>Mitigated by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data integrity &amp; roll-back attacks</td>
<td>TEE security properties</td>
</tr>
<tr>
<td>Timestamp omission attacks</td>
<td>In-enclave checks</td>
</tr>
<tr>
<td>List substitution attacks</td>
<td>In-enclave checks</td>
</tr>
<tr>
<td>TEE reset attacks</td>
<td>Rate-limited by provisioning authority</td>
</tr>
<tr>
<td>TEE side-channel attacks</td>
<td>Ongoing research and/or new TEEs</td>
</tr>
<tr>
<td>CACTI Farms</td>
<td>Cost?</td>
</tr>
</tbody>
</table>
## Security evaluation

<table>
<thead>
<tr>
<th>Adversarial server</th>
<th>Mitigated by</th>
</tr>
</thead>
</table>
| Client tracking    | Group signature scheme  
|                    | Not revealing actual client rates |

<table>
<thead>
<tr>
<th>Adversarial PA</th>
<th>Mitigated by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does not verify remote attestation</td>
<td>Websites can decide which PAs to trust</td>
</tr>
</tbody>
</table>
## Performance evaluation (Latency)

<table>
<thead>
<tr>
<th></th>
<th>ECDSA-Sign</th>
<th>Browser</th>
<th>Pre-Enclave</th>
<th>In-Enclave</th>
<th>Post-Enclave</th>
<th>EPID-Verify</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000 timestamps</td>
<td>6.3 ms</td>
<td>15.2 ms</td>
<td>7.7 ms</td>
<td>181.7 ms</td>
<td>1.0 ms</td>
<td>27.3 ms</td>
<td>239.2 ms</td>
</tr>
<tr>
<td>in 1 list</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4,096 lists with 1</td>
<td>6.3 ms</td>
<td>15.2 ms</td>
<td>1.8 ms</td>
<td>157.4 ms</td>
<td>2.0 ms</td>
<td>27.3 ms</td>
<td>210.0 ms</td>
</tr>
<tr>
<td>timestamp each</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Performance evaluation (Data transfer)

<table>
<thead>
<tr>
<th></th>
<th>Received</th>
<th>Sent</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image-based</td>
<td>140.05 kB</td>
<td>28.97 kB</td>
<td>169.02 kB</td>
</tr>
<tr>
<td>Behavior-based</td>
<td>54.38 kB</td>
<td>26.12 kB</td>
<td>80.50 kB</td>
</tr>
<tr>
<td>CACTI</td>
<td>0.82 kB</td>
<td>1.10 kB</td>
<td>1.92 kB</td>
</tr>
</tbody>
</table>
CACTI: CAPTCHA Avoidance via Client-side TEE Integration

- Using client-side TEEs to provide signals of trustworthiness to websites
- Proof-of-concept implementation within constraints of current TEE hardware
- Possible enabler for new use cases?

Introduced CACTI rate-proof as a versatile primitive providing "express checkout" for legitimate users

Security level is server-configurable, and is always no worse than existing CAPTCHA schemes

Questions?  ercano@uci.edu  nakatsuy@uci.edu