Suppressing Malicious Bot Traffic using an Accurate Human Attester (No Demo)

Muhammad Jamshed (Student), Younghwan Go (Student), and KyoungSoo Park

Department of Electrical Engineering, KAIST
{ajamshed, yhwan}@ndsl.kaist.edu, kyoungsoo@ee.kaist.ac.kr

Malicious bots are widespread and growing in number, posing a serious threat to today's Internet. In March 2011, South Korea saw a massive launch of distributed denial-of-service attacks to major government and portal sites from bots. Bots are not only used for spamming and click frauds, but are also employed in online games to beat other users and to seek unfair gain.

Human attestation is an emerging technology that could potentially eliminate these bot attacks. With an unforgeable human proof, human-attested messages can be safely accepted by the server as bot-free. Unfortunately, existing mechanisms do not accurately bind human activity to the messages, allowing bots to steal the attestation for random contents. [1]

We develop an accurate human message attestation framework for network applications by placing the root of trust on the input devices and a TPM. Our framework binds each key event to the message itself so that smart bots cannot abuse the key event for generating false attestation. For this, we slightly modify the input devices such that they generate a proof (an HMAC-SHA1 hash) for each input event, which is later self-verified.

Our framework provides a guarantee that no attackers can produce valid human attestation signatures unless they run the valid attester code and use physical input devices. It generates the signature that is tightly bound to the message itself. Only when the message is confirmed to be produced from relevant keystrokes or mouse clicks, our framework enables a human attestation signature for the message. Our human attester runs in a sandboxed environment through Flicker [2] using late launch technology. This allows the attester to reliably verify the integrity of the message and attest to human content without interference from malicious software.

The overall attestation procedure is described as follows. A client application records all relevant keycode comprised of input events and their proofs, for a human message. When it needs a human attestation for the message, it starts a late launch environment and executes the attester in a Flicker session. The attester asks the input device to verify the proofs and when they are confirmed to be valid, an embedded TPM signs on the message with its related attributes, and the attester code hash. Finally, the application sends the message with the signature to a remote server over the network, and the server verifies the signature and enforces its own message accepting policy.

TASK	Latency (ms)
Late launch init	130
TPM operations	395
Late launch exit	80
Total	**605**

Figure 1: Human attestation proof generation latency

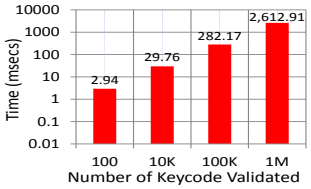

Figure 2: Keycode verfication time for various sizes in attester

We implement our framework on a standard PC with the late launch capability and a system TPM. Our prototype runs on a regular Linux operating system with keyboard/mouse drivers to emulate the input event proof generation and verification. Our result shows 605 milliseconds latency (Figure 1) for human attestation proof generation in typical Web browsing, which is in the range of interactive use. Figure 2 shows that for an email with 10,000 characters, the keycode verification takes less than 30 milliseconds.

References

[1] R. Gummadi, H. Balakrishnan, P. Maniatis, and S. Ratnasamy. Not-a-Bot (NAB): Improving service availability in the face of botnet attacks. In *Proceedings of USENIX NSDI*, 2009.

[2] J. McCune, B. Parno, A. Perrig, M. Reiter, and H. Isozaki. Flicker: An execution infrastructure for TCB minimization. In *Proceedings of EuroSys*, 2008.