Towards Auditing of Control-Flow Integrity

Luke Atherton - Supervised by: Konstantinos Markantonakis

Information Security Group, Smart Card and IoT Security Center
Royal Holloway, University of London

The Smart Card and Internet of Things Security Centre

Objectives

- Investigate existing method providing control-flow integrity;
- Propose a solution for enabling the audit of control-flow integrity.

Introduction

- Control-flow integrity is an important measure of secure software execution;
- Control-flow integrity is a policy which states that the execution flow of an application must follow the control-flow graph generated from the application;
- The problem of enforcing control-flow integrity can be approached from a three different directions: prevention, detection and attestation;
- In this paper, we intend to add a fourth method of enforcing control-flow integrity - audit.We will propose a solution which enables the tracking and storing of control-flow data in audit-friendly reports.

Control-Flow Graphs

Control-flow graphs (CFG) are a method used to formally describe the legitimate paths an application can take during execution. A simple of measure of control-flow integrity is to check whether instructions are processed in an order which abides by the application's CFG.

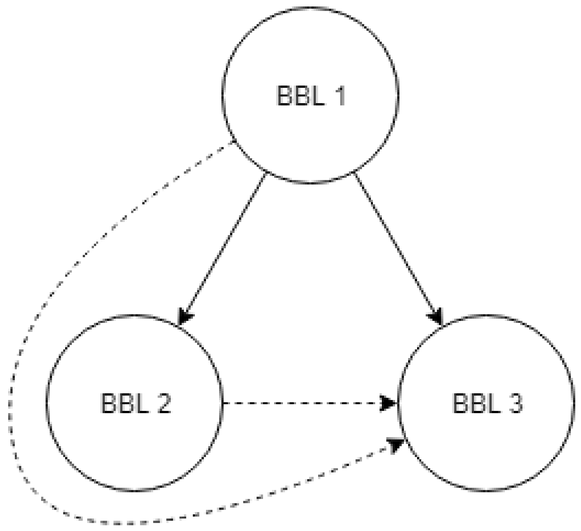


Figure 1:Illegal control-flow

Key Agreement

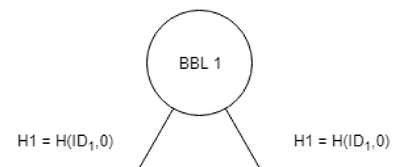
Setup Each user u_i sends a key establishment request, containing its temporary identity TID_i, partial public key q_i, and expiration date/time t_i of its partial public key.

Round 1 Each user u_i verifies that t_j is not out-of-date. Upon successful verification, u_i chooses a random $r_i \in \mathbb{Z}_q^*$, $k_i \in \{0,1\}^k$, and generates a set of ephemeral keys $P_{i,j} = r_i(q_j P + P_0) = r_i(q_j + s)P$ for $1 \leq j \leq n$ and $j \neq i$. Each user u_i then broadcasts the set of $P_{i,j}$ along with $H_3(k_i)$.

Round 2 Upon reception of $H_3(k_j)$ and $P_{j,i}$, each user u_i computes $sid_i^w = H_3(k_1)||...||H3(k_n)$. Each user u_i then generates the set of $t_{j,i} = e(P_{j,i}, D_i)^{x_i} P_j^{r_i} = g^{r_j x_i + r_i x_j}$, $V_{j,i} = H_2(t_{j,i}||sid_i^w)$, and $K_{j,i} = V_{j,i} \oplus k_i$. The set of $K_{j,i}$ is broadcast.

Key generation Upon reception of $K_{i,j}$, u_i computes $\tilde{k}_j = V_{j,i} \oplus K_{i,j}$ and checks whether $H_3(\tilde{k}_j) = H_3(k_j)$ is valid. Upon successful verification, each user u_i generates the session key,

$$sk_i^w = H_3(k_1||...||k_n||sid_i^w||pid_i^w)$$



Evaluation

- The *Syther* tool will be used to formally analyze the protocol;
- The experimental setup will consist of a set of Raspberry Pi 2 Model B+ System-on-Chip, communicating via wireless LAN interface.
- The number of users n taking part in the key agreement is a critical parameter to the computational cost and its effects will be analyzed. We expect the bilinear pairings to be the most expensive operations.

Criteria	Protocol
Number of rounds	2
Number of modular exponentiations	$n-1$
Number of bilinear pairings	$n-1$
Number of elliptic curve scalar point multiplication	$3n-2$
Communication overhead	$2(n-1)\|P\|$

Table 1:Complexity analysis

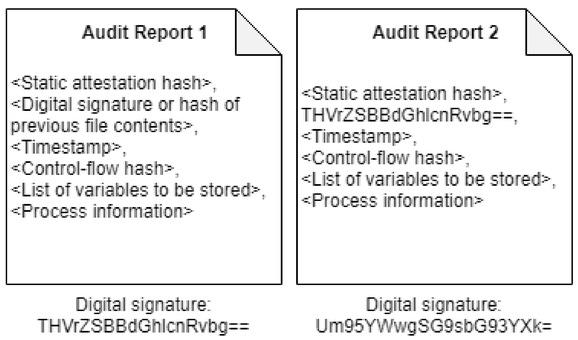


Figure 3:Audit files

Benefits

The proposed solutions enables the a new method of handling control-flow within embedded systems:

- Historic evidence of control-flow;
- Binding of variables to control-flow snapshot;

Conclusion

The proposed protocol enables a fleet of UAVs to derive a unique symmetric key. It captures the following security properties: mutual authentication, mutual key agreement, joint key control, key freshness, entity revocation, non-repudiation, forward secrecy, known-key security, and conditional privacy. Since the protocol is certificateless-based, the necessity for a public key infrastructure is eliminated, as well as the key escrow problem. This research paper will be submitted to the *DASC 2018* conference.

References

Contact Information

- Web: https://scc.rhul.ac.uk/
- Email: luke.atherton.2018@live.rhul.ac.uk
- Twitter: @LucialHz