One-pager 7

Martin Vassor

1 Introduction

1.1 Objectives

The designed device guarantees both secrecy and integrity of sensitive data. During a computation, it ensures correctness of sensitive parts of the computation with respect to the machine code provided (i.e. the provided code is indeed what is executed).

Threat model: The *host* system can read an modify all local data.

1.2 General design

Sensitive programs are stored on the device. During execution, insensitive sections are computed on the *host* system, and sensitive section on the *guest*. An untrusted driver is required on the *host* system to receive insensitive data and instructions. The design is conservative: if the driver is compromised, the safety properties still holds.

2 Determining sensitive data

Sensitive data is statically defined. The programmer statically declares which data is considered sensitive (noted $ss(data)$). Each instruction is either *sensitive-preserving* ($sp(instr)$) or *sensitive-masking* ($sm(instr)$). Intuitively an instruction is sensitive-masking if it hides the sensitive data. It may be the case that (1) the language is high-level (no direct pointer manipulation, etc.); and (2) some insensitive data is considered sensitive.

Inference rules to statically determine which data is sensitive ($sd(data)$) are the following:

$$\frac{ss(data)}{sd(data)}\ (S.Decl) \qquad \frac{data = instr(op1, op2) \quad sm(instr)}{\neg sd(data)}\ (S.Mask)$$

$$\frac{data = instr(op1, op2) \quad sp(instr) \quad sd(op1)}{sd(data) \quad sd(op2)}\ (S.Pres1)$$

$$\frac{data = instr(op1, op2) \quad sp(instr) \quad sd(op2)}{sd(data) \quad sd(op1)}\ (S.Pres2)$$

An operation must be executed on the *guest* device if one of $sd(op1)$, $sd(op2)$ or $sd(res)$ holds. Similarly, sensitive data is stored on the *guest* device.

3 Bootstrapping, running a program, and I/O

The device initiate the execution of any sensitive program. It sends insensitive data and parts of code to the *host* system, and collect the result for sensitive sections. This requires each sensitive program to be either pre-compiled in a trusted environment; or to embed a trusted compiled compiler in the device, which can securely compile programs.

As the *host* system is fully compromised, the *guest* device must have its own I/O peripherals for sensitive I/O operations. However, if the threat model is relaxed, it might use the *host* one's.

4 Argument for guarantees

Secrecy Initially, all sensitive data are stored in the *guest* device. During the execution, only insensitive data are sent to the *host* system. Also, dynamically created sensitive data only results from rules ($S.Pres1$) and ($S.Pres2$), which one of the operand is sensitive, i.e. it is executed on the *guest* device. Hence all sensitive data is located in the *guest* device during the computation.

Correctness of computation From rules ($S.Pres1$) and ($S.Pres2$), any operation producing sensitive data marks its operand as sensitive, hence is computed in the *guest* device. The property hence hold by induction.

Integrity Trivial after the two above properties.

5 Conclusion

The proposed device statically determine which data is sensitive and which is not. Based on this distinction, the compiler determines what must be stored or performed on the *guest* device.

Word count:
```
$pdftotext OP7_vassor.pdf - | wc -w
497
```