1 Bootstrapping Trust in Commodity Computers

1.1 Introduction

Bootstrapping trust in commodity computers can be done by conveying information about the computer's current execution state to interested parties (the user). Research in trusted computing can guide the development of new hardware-supported security features.

1.2 Techniques for Recording Platform State

The state of the computer is a combination of it's hardware configuration and the code that it has executed. Code identity needs to be established before being able to make a trust decision. Identifying code is often done with cryptographic hashes. Code currently in control needs to be identified to start gaining trust in the system, also code that has previously been in control and could have made an impact on the environment of the system. Code is measured when it is about to be executed, it is still in it's canonical form at that point. These measurements of code create the chain of trust. The code identity records need to be secured themselves against attackers. Hash chains are an effective way of doing this possibly protecting against both privilege escalation and hand-off attacks.

Dynamic properties are also important to take into account. These can be intended control flow, preserved integrity of data structures or information flow control. These properties are less elementary to the trust of the system than the code identification. This means that code identification could benefit more from hardware support than the dynamic properties would.

1.3 Can We Use Platform Information Locally?

Secure boot is one way to make sure that the platform state is used to prevent malicious code from being loaded. This method verifies that the code identification is certified and prevents execution if it is not. A remote party is not able to identify the state in which the computer is loaded, even if they are able to figure out that the system has booted securely.

Trusted Platform Modules (TPM) can be used to seal the storage to ensure that the bootsection cannot be modified when the system is in a certain state.

1.4 Can We Use Platform Info. Remotely?

Attestation can be done by a verifier if they have access to the public key linked to the private key of the root of trust of the attestor. It is important to make sure that reboot attacks are mitigated when using this method, this can be achieved by communication through a secure tunnel or channel. Privacy of

the platforms can be increased by using a third party (Privacy Certificate Authority) that maps pseudonyms to the real devices and thus makes sure the identities can remain secret if necessary. This centralised solution may not be the best, there are also decentralized alternatives like Decentralised Anonymous Attestation.

1.5 How Do We Make Sense of Platform State?

Computer systems are very complex with millions of lines of code which are hard to verify or check for bugs. This amount of verification work can be reduced by only focussing on security relevant code. This code should have privilege layering and enforce isolation between itself and less privileged code. When using VMM's the problem remains because lots of code runs before the VMM meaning it has higher privileges and can thus tamper with the execution of the VMM itself, a solution to this problem has been found named late launch which resets the state of the platform to a known state and by doing so shortens the chain of trust. A different approach is to convert the information into higher level properties of the software that should help to judge about the trust.

1.6 Roots of Trust

1.7 Validating the Process

1.8 Applications

1.9 Human Factors & Usability

1.10 Limitations

Checking code before it is about to be run is much easier than checking it while it is executing but this approach is also less robust. A buffer overflow vulnerability for instance is exploited during run time and will never be found with static attestation. Other problems arise when the attacker has physical access to the devices, tampering with the reset pin of a TPM can undermine the security it offers.

1.11 Future Directions and Open Questions

1.12 Additional Reading

1.13 Conclusions