Lightweight and Flexible Trust Assessment Modules for the Internet of Things

Jan Tobias Mühlberg, Job Noorman and Frank Piessens

jantobias.muehlberg@cs.kuleuven.be
iMinds-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

QA&Test @ Bilbao, October 2015

COSIC and DistriNet: Who we are.

COSIC (Bart Preneel, Ingrid Verbauwhede)

- Cryptographic primitives
  RIJNDAEL (AES), LANE (SHA-3 candidate)
- Secure and compact hardware design
  SPONGENT (lightweight hash), Side-channel attacks

DistriNet (Frank Piessens)

- Low-level vulnerabilities and countermeasures
  Still very relevant in the IoT
- Protected module architectures
  Software isolation with a minimal TCB
- Fully abstract/secure compilation
  Enable security reasoning at high-level languages

Lightweight and Flexible Trust Assessment Modules for the **Internet of Things**

Jan Tobias Mühlberg, Job Noorman and Frank Piessens

jantobias.muehlberg@cs.kuleuven.be
iMinds-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

QA&Test @ Bilbao, October 2015

Motivation: Security of IoT Nodes

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]

Motivation: Security of IoT Nodes

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]

Safety and security?

- No MMU, no hierarchical protection domains, etc.
- Successful attacker has full control over a node:
  - Modify all code and data
  - Perform I/O

Motivation: Security of IoT Nodes

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]

Safety and security?

- No MMU, no hierarchical protection domains, etc.
- Successful attacker has full control over a node:
  - Modify all code and data
  - Perform I/O
  - DoS, forge sensor readings or node identity

Motivation: Security of IoT Nodes

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]

Safety and security?

- No MMU, no hierarchical protection domains, etc.
- Successful attacker has full control over a node:
  - Modify all code and data
  - Perform I/O
  - DoS, forge sensor readings or node identity
- Even without an attacker: bugs and software ageing

Motivation: Security of IoT Nodes

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]

Safety and security?

- No MMU, no hierarchical protection domains, etc.
- Successful attacker has full control over a node:
  - Modify all code and data
  - Perform I/O
  - DoS, forge sensor readings or node identity
- Even without an attacker: bugs and software ageing
- Trustworthiness of a node is hard to assess! Testing? Formal verification? Observation?

Motivation: Security of IoT Nodes

TI MSP430: designed for low cost and low power consumption

- Runs 4.5 years on a single AAA cell and almost 13 years on an AA battery [Sea08]

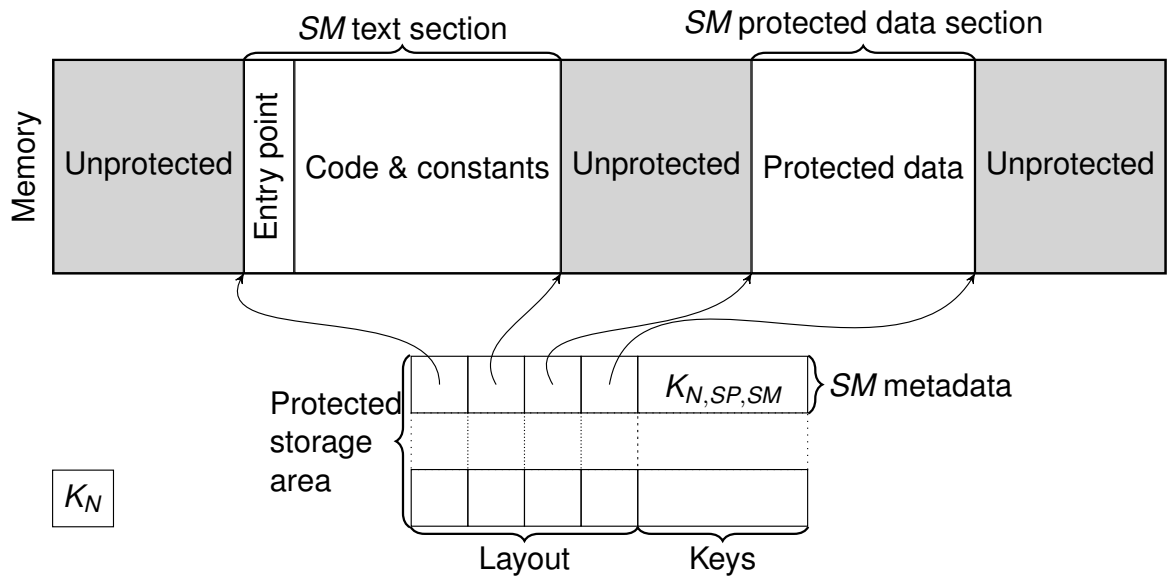
Safety and security?

- No MMU, no hierarchical protection domains, etc.
- Successful attacker has full control over a node:
  - Modify all code and data
  - Perform I/O
  - DoS, forge sensor readings or node identity
- Even without an attacker: bugs and software ageing
- Trustworthiness of a node is hard to assess!
  Testing? Formal verification? Observation?
- Protected Module Architectures can help
  (Intel SGX, ARM TrustZone, SMART, TrustLite, Sancus)

Motivation: Sancus

Sancus [NAD+13] enables strong isolation, attestation and communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

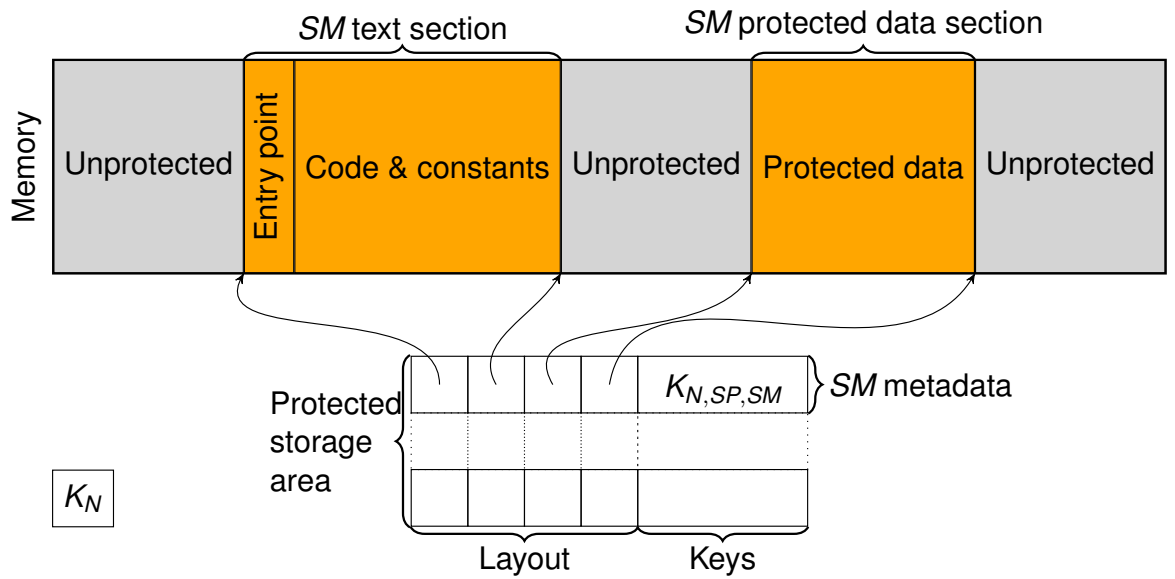


Motivation: Sancus

Sancus [NAD+13] enables strong isolation, attestation and communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Public and protected sections

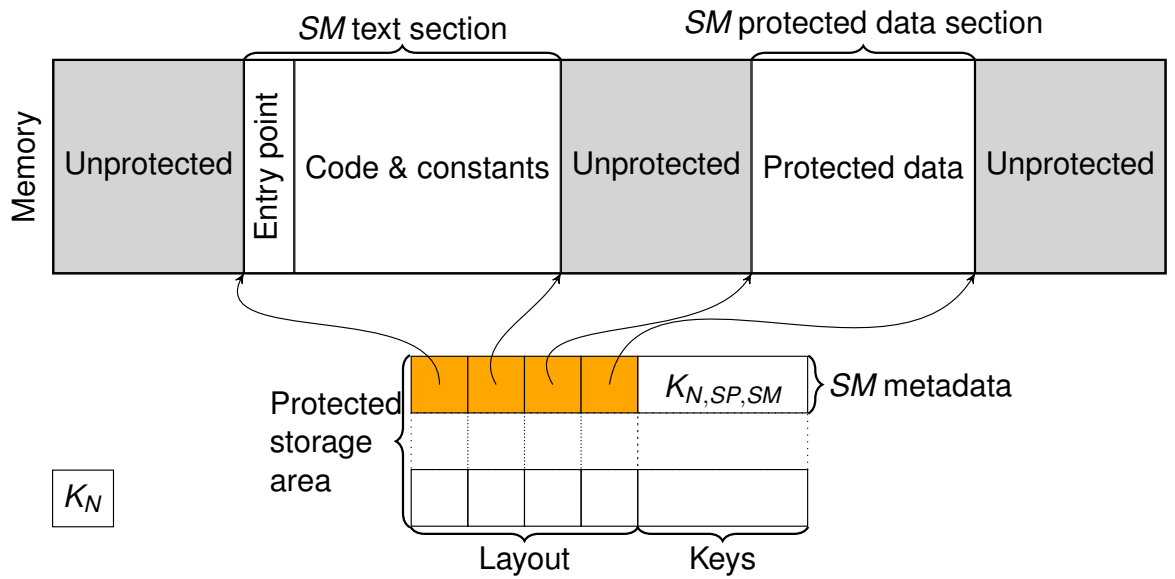


Motivation: Sancus

Sancus [NAD+13] enables strong isolation, attestation and communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Module layout

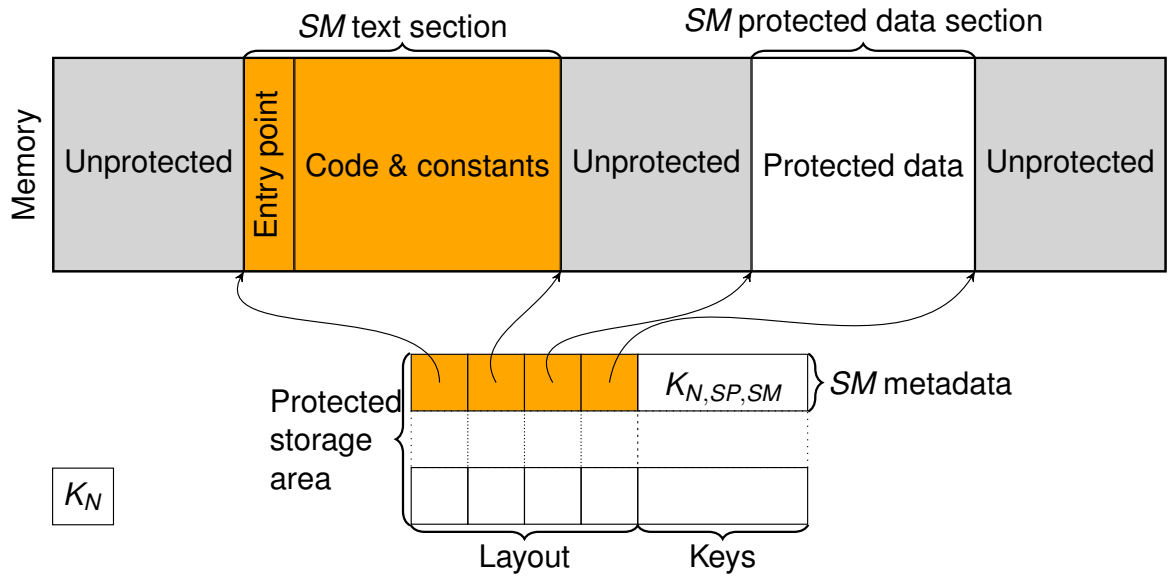


Motivation: Sancus

Sancus [NAD+13] enables strong isolation, attestation and communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Module identity

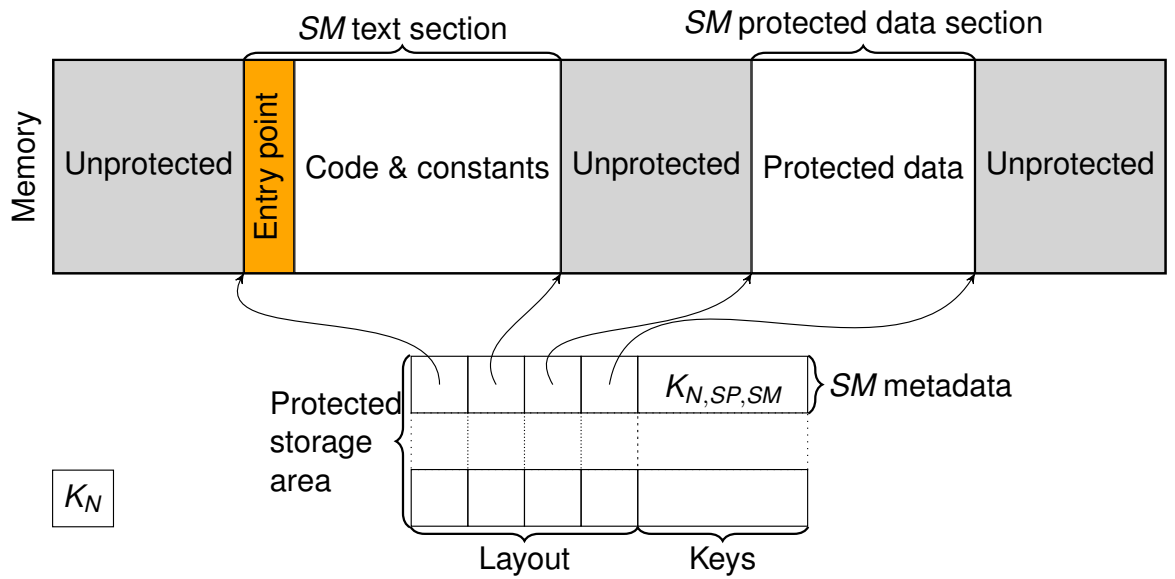


Motivation: Sancus

Sancus [NAD+13] enables strong isolation, attestation and communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Module entry point

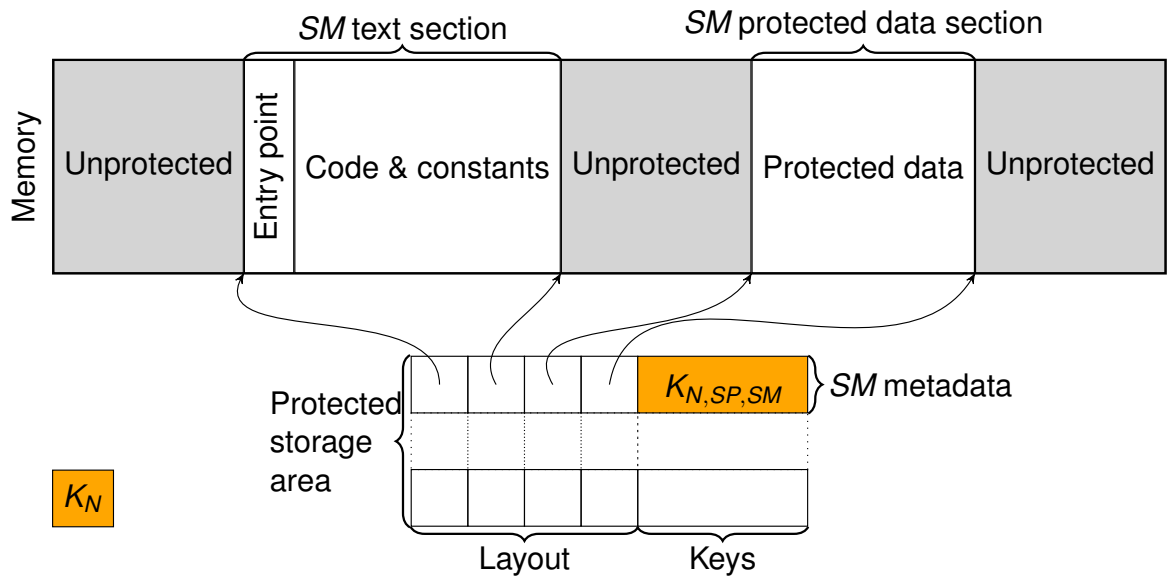


Motivation: Sancus

Sancus [NAD+13] enables strong isolation, attestation and communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures

Module keys



Motivation: Sancus

Sancus [NAD+13] enables strong isolation, attestation and communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures
- Provides efficient cryptographic primitives and key handling
- Reference implementation based on the openMSP430

Motivation: Sancus

Sancus [NAD+13] enables strong isolation, attestation and communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures
- Provides efficient cryptographic primitives and key handling
- Reference implementation based on the openMSP430

Some drawbacks:

- Isolation vs. shared memory communication [BNMP15]
- Re-implementing an existing set of applications as SMs is often not straight-forward

Motivation: Sancus

Sancus [NAD+13] enables strong isolation, attestation and communication for embedded software components:

- Implements Program Counter Based Access Control [SPP10] for Software Modules (SMs) on single-address-space architectures
- Provides efficient cryptographic primitives and key handling
- Reference implementation based on the openMSP430

Some drawbacks:

- Isolation vs. shared memory communication [BNMP15]
- Re-implementing an existing set of applications as SMs is often not straight-forward

Can we use Sancus SMs to implement light-weight and secure inspection components that integrate seamlessly with existing deployment scenarios?

Trust Assessment Modules

Idea

- Securely deploy a protected inspection module to assess the state of an IoT node

Trust Assessment Modules

Idea

- Securely deploy a protected inspection module to assess the state of an IoT node
- Modules can be configured to perform inspection tasks autonomously or upon request
- Modules can be securely unloaded or exchanged
- Sancus crypto and key management facilitates secure and authenticated communication with operator or trust management system

Trust Assessment Modules

Idea

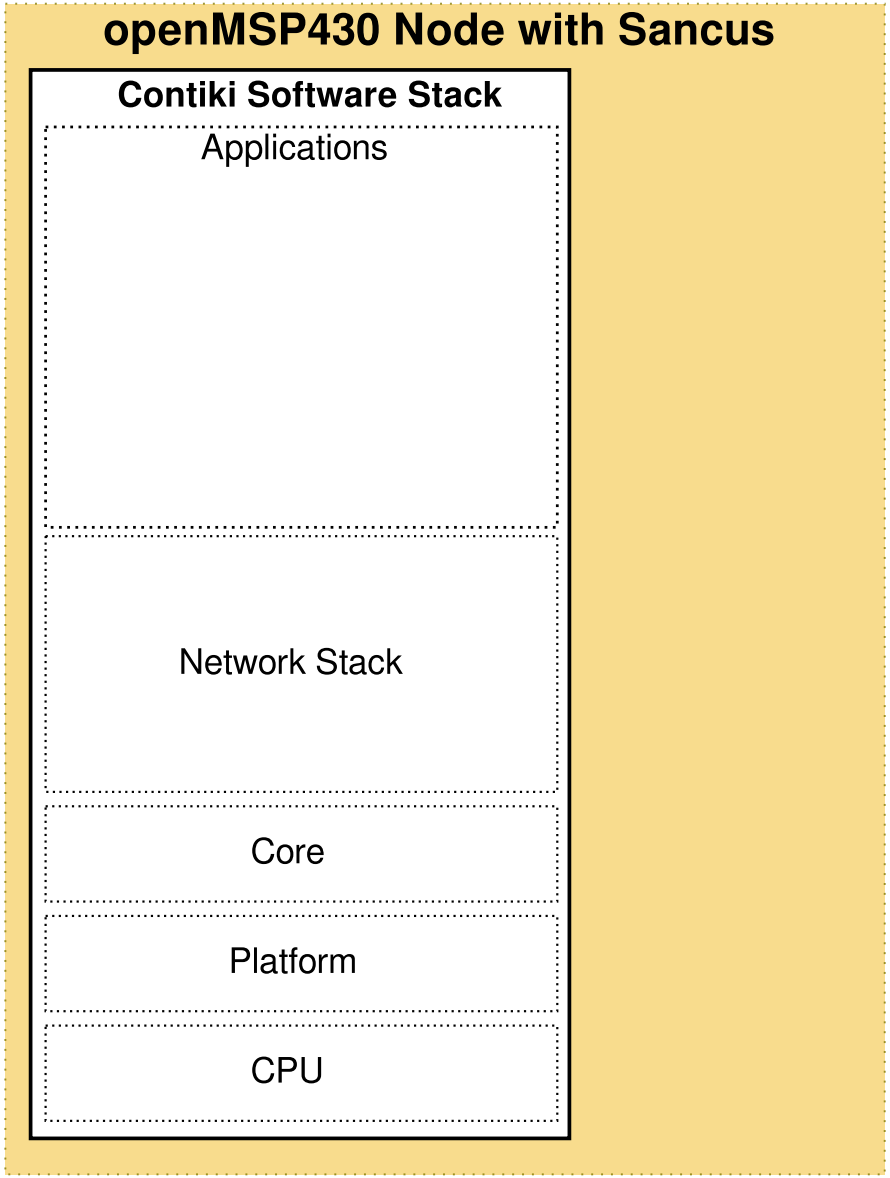
- Securely deploy a protected inspection module to assess the state of an IoT node
- Modules can be configured to perform inspection tasks autonomously or upon request
- Modules can be securely unloaded or exchanged
- Sancus crypto and key management facilitates secure and authenticated communication with operator or trust management system

→ Flexible and authenticated node inspection

→ No or minimal changes in existing deployment scenarios

→ Easy reconfiguration impedes attacker adaptation

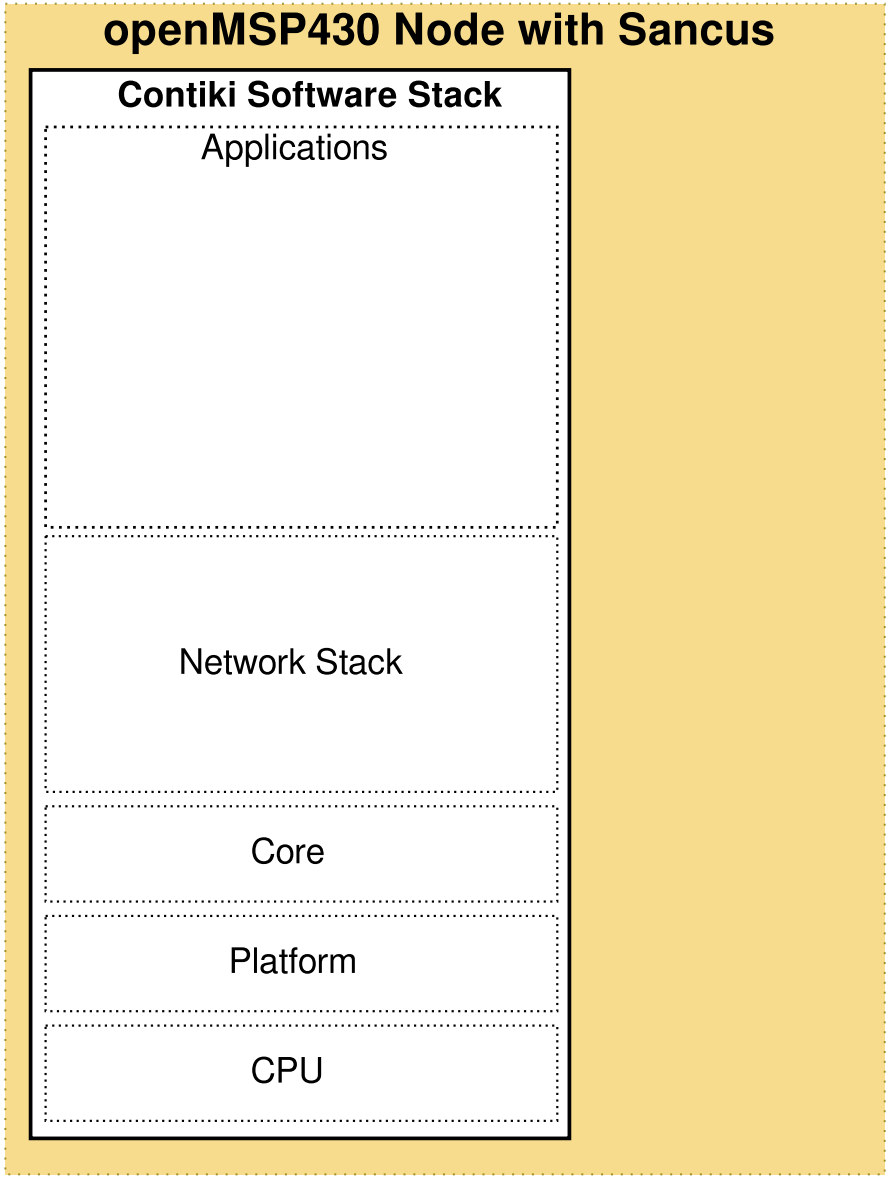
Secure Module Deployment

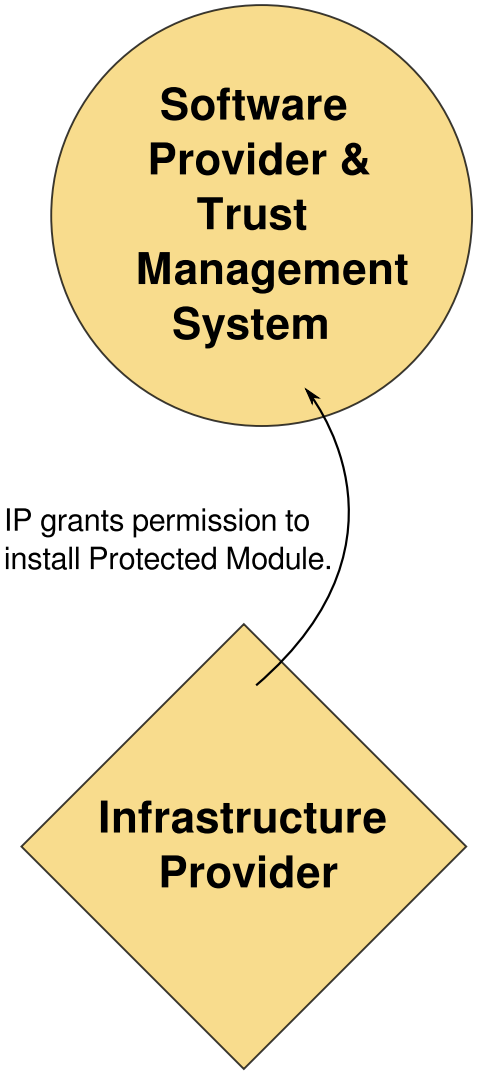


Software Provider & Trust Management System

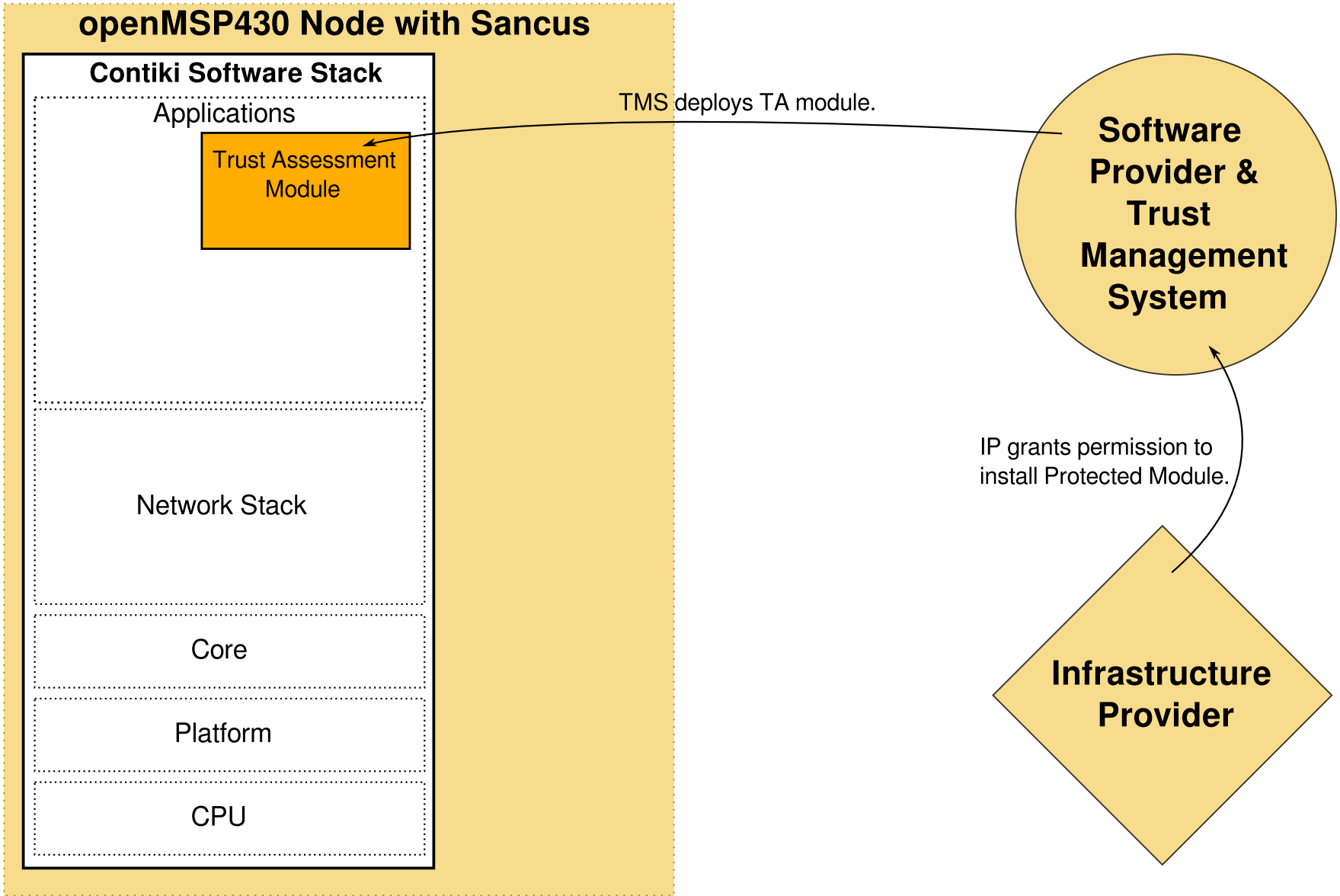
Infrastructure Provider

Secure Module Deployment

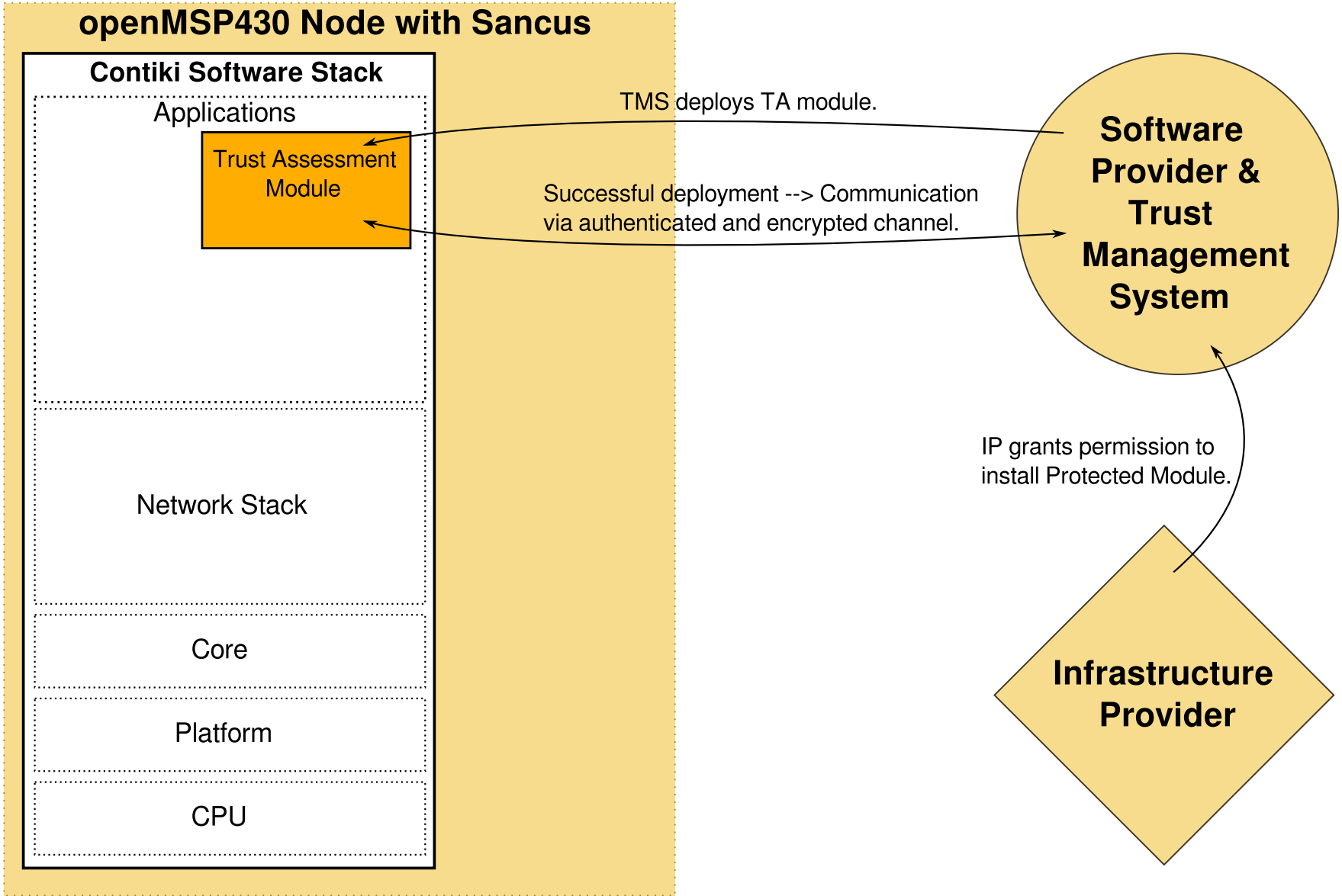




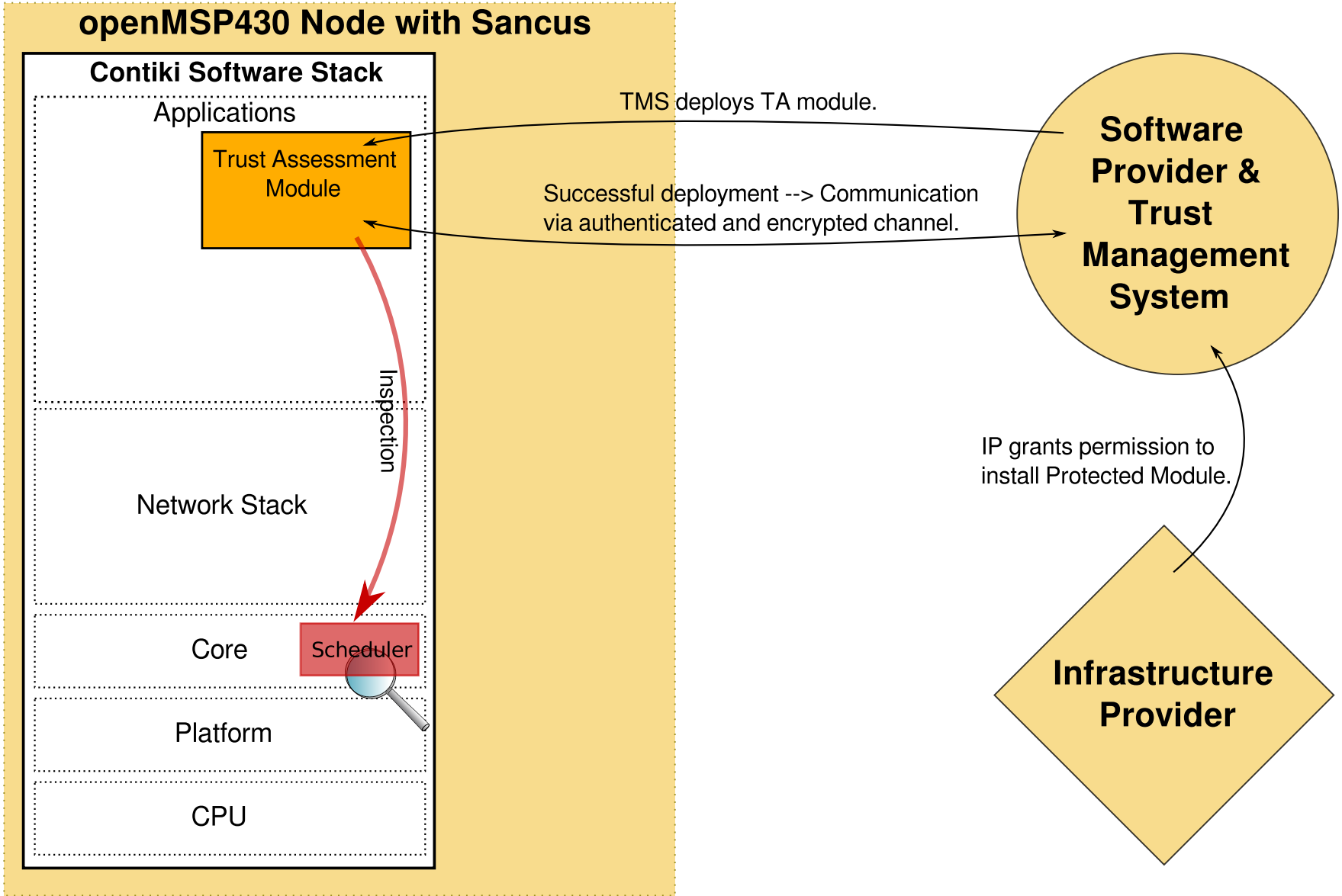
Secure Module Deployment



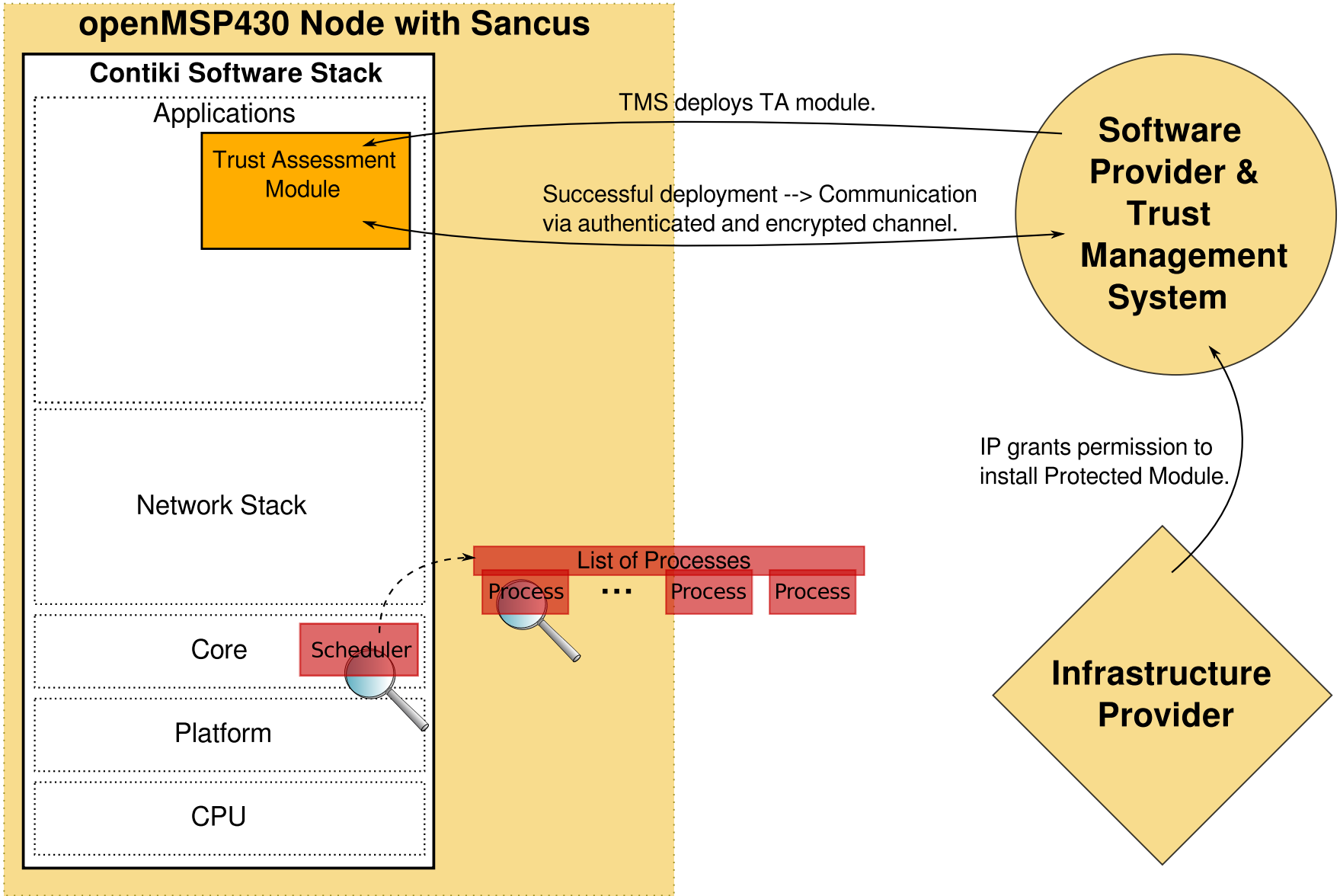
Secure Module Deployment



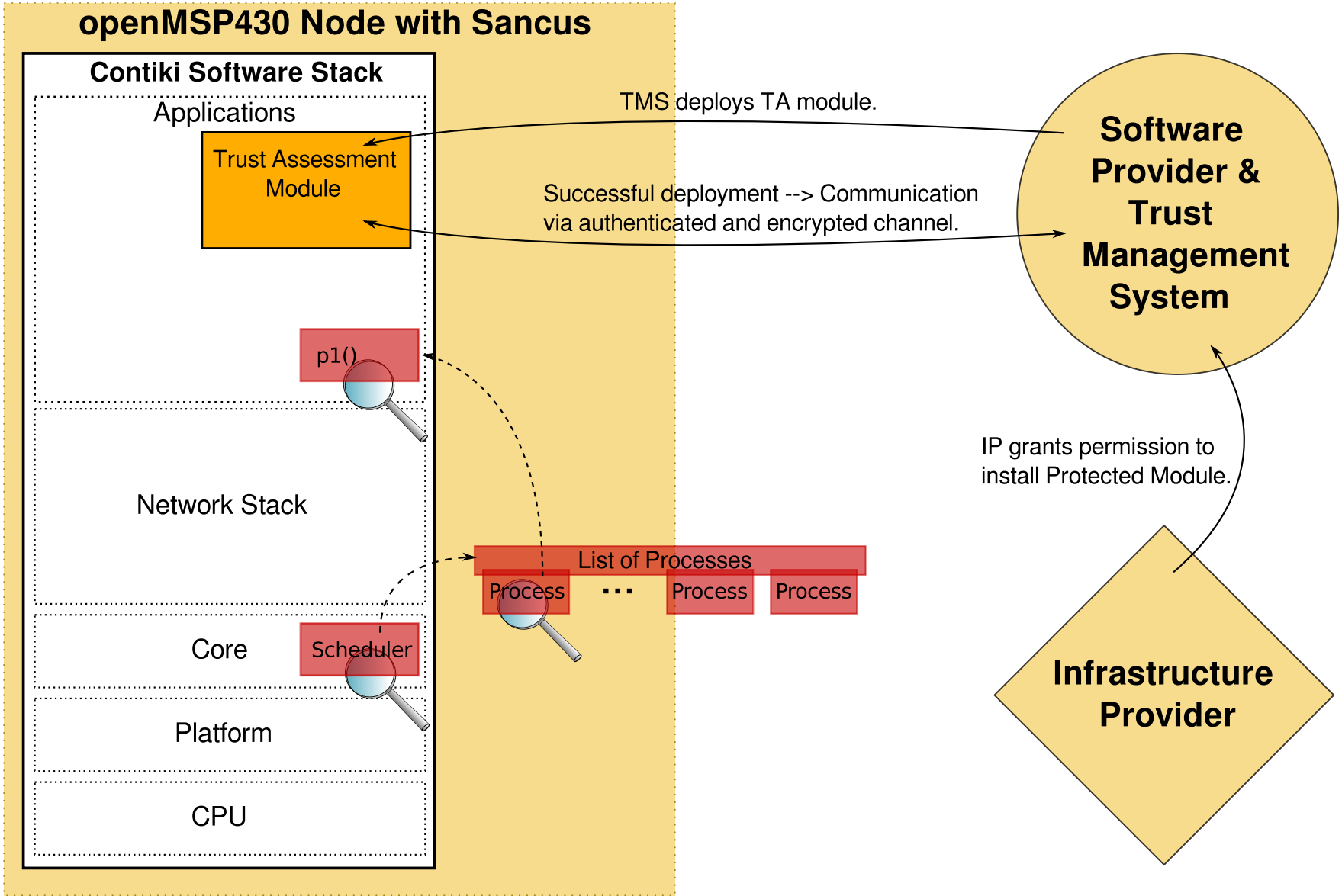
Inspection Targets



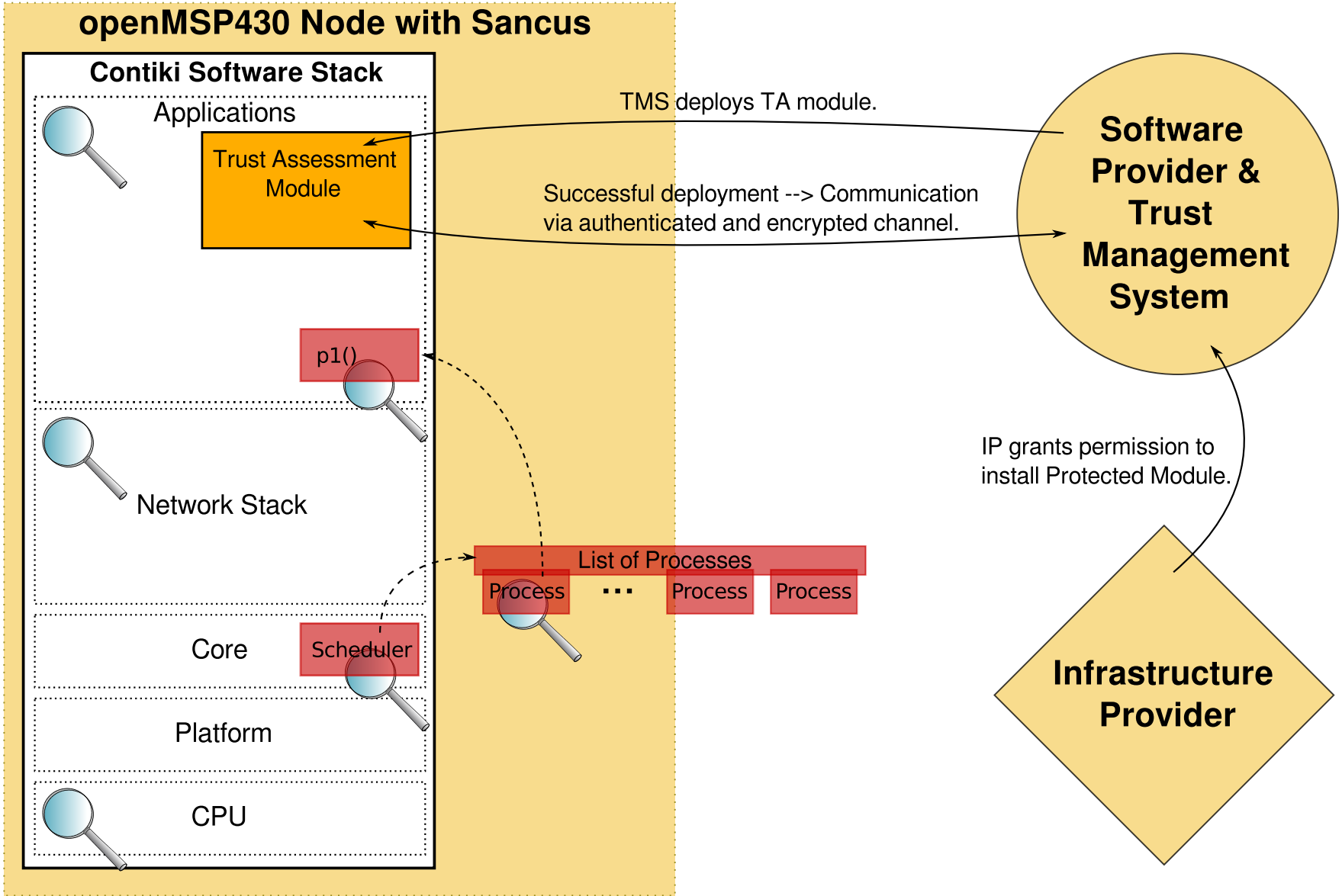
Inspection Targets



Inspection Targets



Inspection Targets



Trust Indicators

What to Inspect?

Trust Indicators

What to Inspect?

- Code Integrity

Trust Indicators

What to Inspect?

- Code Integrity
- OS Data Structures
- Application Data Structures

Trust Indicators

What to Inspect?

- Code Integrity
- OS Data Structures
- Application Data Structures
- Resource Availability
- Event Occurrence and Timing

Trust Indicators

What to Inspect?

- Code Integrity
- OS Data Structures
- Application Data Structures
- Resource Availability
- Event Occurrence and Timing
- Combined Indicators

Trust Indicators

What to Inspect?

- Code Integrity
- OS Data Structures
- Application Data Structures
- Resource Availability
- Event Occurrence and Timing
- Combined Indicators

A Realistic Scenario

Trust Assessment Module monitors and reports

- list of running processes on a Contiki node
- code integrity of the main function of each process
- number of invocations and time of last invocation

Trust Indicators

What to Inspect?

- Code Integrity
- OS Data Structures
- Application Data Structures
- Resource Availability
- Event Occurrence and Timing
- Combined Indicators

A Realistic Scenario

Trust Assessment Module monitors and reports

- list of running processes on a Contiki node
- code integrity of the main function of each process
- number of invocations and time of last invocation

Attacker process periodically

- modifies process list
- modifies code section

Evaluation

Size and execution time of different trust assessment components on an MSP430 running at 20 MHz: 1 cycle corresponds to 50 ns

Function	**Size** in Bytes	**Runtime** in Cycles	**Description**
`TACoreEnable`	58	236,440	Enables module protection and initiates key generation
`TAMainFunc`	430	578 73,678	Main function, initialisation . . . validation run (5 processes, 9 integrity checks)
`TARegisterInvar`	402	1,242 10,762 19,930	Stores meta-data and MACs of 32 B . . . 199 B . . . 399 B
`TACheckInvars`	498	69,659	Checks integrity of 9 address ranges (1833 B)
`TAAddProcess`	568	$\leq$ 18,374	Shadows an entry from the process list and determines length of process function
`TACheckProcesses`	288	2,371	Checks shadowed process data against process list (5 processes)
`TASecureCallProcess`	392	266 $\leq$ 731	Process invocation with no logging . . . logs time and and number of invocations
`TAInvarsStatus`	202	10,254	Encrypts and signs meta-data on integrity-checked code and data (160 B + 16 B nonce)
`TAProcessStatus`	202	17,488	Encrypts and signs meta-data on running processes (320 B + 16 B nonce)
total	3,742	n/a	Code (`.text`) and data (part of `.bss`)

Sancus on Spartan-6: +50 % LUTs and registers, +6 % power consumption

236,440 cycles $\approx$ 0.012 s

Conclusions

Trust Assessment Modules for the IoT

- Based on Sancus, a hardware-only PMA
- Secure and authenticated node inspection
- No or minimal changes to existing code
- Easy reconfiguration for flexibility and to impede attacker adaptation
- Acceptable memory footprint and performance

Conclusions

Trust Assessment Modules for the IoT

- Based on Sancus, a hardware-only PMA
- Secure and authenticated node inspection
- No or minimal changes to existing code
- Easy reconfiguration for flexibility and to impede attacker adaptation
- Acceptable memory footprint and performance

Use Cases

- Node inspection and trust assessment

Conclusions

Trust Assessment Modules for the IoT

- Based on Sancus, a hardware-only PMA
- Secure and authenticated node inspection
- No or minimal changes to existing code
- Easy reconfiguration for flexibility and to impede attacker adaptation
- Acceptable memory footprint and performance

Use Cases

- Node inspection and trust assessment: can be integrated with trust management infrastructure

Conclusions

Trust Assessment Modules for the IoT

- Based on Sancus, a hardware-only PMA
- Secure and authenticated node inspection
- No or minimal changes to existing code
- Easy reconfiguration for flexibility and to impede attacker adaptation
- Acceptable memory footprint and performance

Use Cases

- Node inspection and trust assessment: can be integrated with trust management infrastructure
- Detect modifications, malware, effects of software aging
- Secure remote debugging of IoT nodes
- Can be implemented using other PMAs that provide isolation and attestation

Conclusions

Trust Assessment Modules for the IoT

- Based on Sancus, a hardware-only PMA
- Secure and authenticated node inspection
- No or minimal changes to existing code
- Easy reconfiguration for flexibility and to impede attacker adaptation
- Acceptable memory footprint and performance

Use Cases

- Node inspection and trust assessment: can be integrated with trust management infrastructure
- Detect modifications, malware, effects of software aging
- Secure remote debugging of IoT nodes
- Can be implemented using other PMAs that provide isolation and attestation

Future Work?

Related Work

Alternatives to Sancus

- Server/Desktop: Intel SGX [MAB+13], ARM TrustZone [AF04], TrustVisor [MLQ+10], Fides [SP12]
- Embedded: SMART [EFPT12] & TrustLite [KSSV14]

Related Work

Alternatives to Sancus

- Server/Desktop: Intel SGX [MAB+13], ARM TrustZone [AF04], TrustVisor [MLQ+10], Fides [SP12]
- Embedded: SMART [EFPT12] & TrustLite [KSSV14]

Trust Assessment on Desktop & Server Systems

- By means of specialised hardware: Copilot [PJFMA04] & Gibraltar [BGI11]
- Kernel extensions & hypervisors: HeapSentry [NPJ13], SecVisor [SLQP07], HyperForce [GNMJ12], etc.

Related Work

Alternatives to Sancus

- Server/Desktop: Intel SGX [MAB+13], ARM TrustZone [AF04], TrustVisor [MLQ+10], Fides [SP12]
- Embedded: SMART [EFPT12] & TrustLite [KSSV14]

Trust Assessment on Desktop & Server Systems

- By means of specialised hardware: Copilot [PJFMA04] & Gibraltar [BGI11]
- Kernel extensions & hypervisors: HeapSentry [NPJ13], SecVisor [SLQP07], HyperForce [GNMJ12], etc.

Trust Management in WSN

- Focuses on observable behaviour of nodes, i.e., communication and plausibility of sensor readings [FGRL07, GMS15, LRAFG10]

Thank you! Questions?

http://distrinet.cs.kuleuven.be/software/sancus/

Further reading: “Lightweight and Flexible Trust Assessment Modules for the Internet of Things.” Mühlberg et al., ESORICS 2015, LNCS vol. 9326, pages 503–520, Springer.

References I

T. Alves and D. Felton.

Trustzone: Integrated hardware and software security.
ARM white paper, 3(4):18–24, 2004.

A. Baliga, V. Ganapathy, and L. Iftode.

Detecting kernel-level rootkits using data structure invariants.
IEEE Transactions on Dependable and Secure Computing, 8:670–684, 2011.

J. V. Bulck, J. Noorman, J. T. Mühlberg, and F. Piessens.

Secure resource sharing for embedded protected module architectures.
In *WISTP '15*, LNCS, Heidelberg, 2015. Springer.
To appear.

K. Eldefrawy, A. Francillon, D. Perito, and G. Tsudik.

SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust.
In *NDSS 2012, 19th Annual Network and Distributed System Security Symposium, San Diego, USA*, 2012.

M. Fernandez-Gago, R. Roman, and J. Lopez.

A survey on the applicability of trust management systems for wireless sensor networks.
In *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2007. SECPerU 2007. Third International Workshop on*, pp. 25–30, 2007.

J. Granjal, E. Monteiro, and J. S. Silva.

Security in the integration of low-power wireless sensor networks with the internet: A survey.
Ad Hoc Networks, 24, Part A(0):264–287, 2015.

References II

F. Gadaleta, N. Nikiforakis, J. T. Mühlberg, and W. Joosen.

HyperForce: Hypervisor-enForced execution of security-critical code.
In *IFIP SEC 2012*, vol. 376 of *IFIP Advances in Information and Communication Technology*, pp. 126–137, Heidelberg, 2012. Springer.

P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan.

Trustlite: A security architecture for tiny embedded devices.
In *Proceedings of the Ninth European Conference on Computer Systems*, EuroSys '14, pp. 10:1–10:14. ACM, 2014.

J. Lopez, R. Roman, I. Agudo, and C. Fernandez-Gago.

Trust management systems for wireless sensor networks: Best practices.
Comput. Commun., 33(9):1086–1093, 2010.

F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar.

Innovative instructions and software model for isolated execution.
In *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pp. 10:1–10:1. ACM, 2013.

J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig.

Trustvisor: Efficient tcb reduction and attestation.
In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pp. 143–158. IEEE, 2010.

J. Noorman, P. Agten, W. Daniels, R. Strackx, A. Van Herrewege, C. Huygens, B. Preneel, I. Verbauwhede, and F. Piessens.

Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base.
In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pp. 479–494, Berkeley, CA, USA, 2013. USENIX Association.

References III

N. Nikiforakis, F. Piessens, and W. Joosen.

HeapSentry: Kernel-assisted protection against heap overflows.
In *Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 7967 of *Lecture Notes in Computer Science*, pp. 177–196. Springer, 2013.

N. L. Petroni Jr, T. Fraser, J. Molina, and W. A. Arbaugh.

Copilot-a coprocessor-based kernel runtime integrity monitor.
In *USENIX Security Symposium*, pp. 179–194. USENIX Association, 2004.

M. Seaman.

Powering an MSP430 from a single battery cell.
Technical Report SLAA398, Texas Instruments, 2008.

A. Seshadri, M. Luk, N. Qu, and A. Perrig.

SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes.
In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pp. 335–350. ACM, 2007.

R. Strackx and F. Piessens.

Fides: Selectively hardening software application components against kernel-level or process-level malware.
In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pp. 2–13, New York, NY, USA, 2012. ACM.

R. Strackx, F. Piessens, and B. Preneel.

Efficient isolation of trusted subsystems in embedded systems.
In *Security and Privacy in Communication Networks*, vol. 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 344–361. Springer, 2010.

Secure Module Deployment