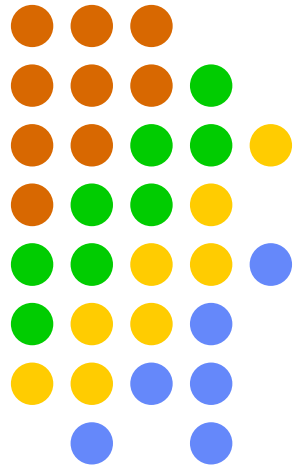
An Overview of Palladium

Brian A. LaMacchia

Software Architect

Windows Trusted Platform Technologies



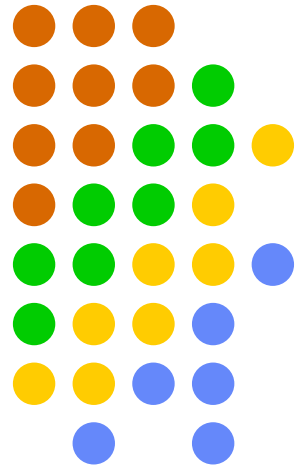
Acknowledgements

- Key contributors to the Palladium initiative at Microsoft include:
  - Peter Biddle
  - John de Treville
  - Paul England
  - Butler Lampson
  - John Manferdelli
  - Marcus Peinado
  - Bryan Willman

Agenda

- Introduction and Motivation
- Architecture
  - New Security Features
- Policy Issues
- Summary/Q&A

Introduction & Motivation



What is Palladium?

- Palladium (Pd) is a set of new security-oriented capabilities in Windows
  - Enabled by new hardware
- Goal is to “protect software from software”
  - Defend against malicious software running in Ring 0
- Four categories of new security features
  - Sealed storage
  - Attestation
  - Curtained memory
  - Secure input and output

Trusted Open Systems

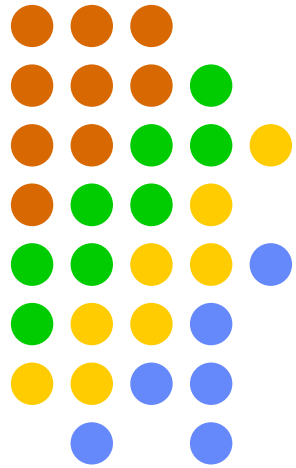
- Our OSs are designed for:
  - Features
  - Performance
  - Plug-ability/Openness
    - Applications
    - Drivers
    - Core OS components
  - Ease of use, and
  - Security

    Contrast this with the design of a smartcard OS

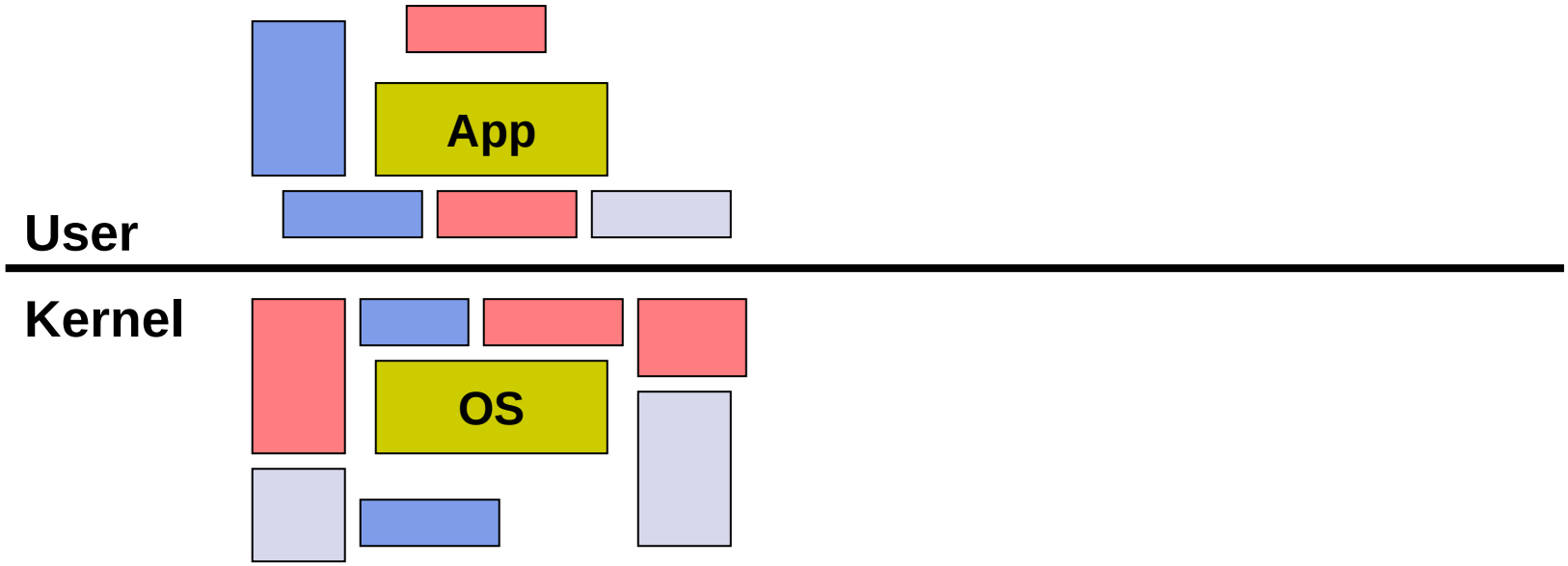
Nightmare Scenarios

- A virus/Trojan that launches something worse than a denial of service attack:
  - Trades a random stock (for mischief or profit)
  - Posts tax-records to a newsgroup
  - Orders a random book from Amazon.com
  - Grabs user/password for the host/web-sites and posts them to a newsgroup
  - Posts personal documents to a newsgroup

Architecture



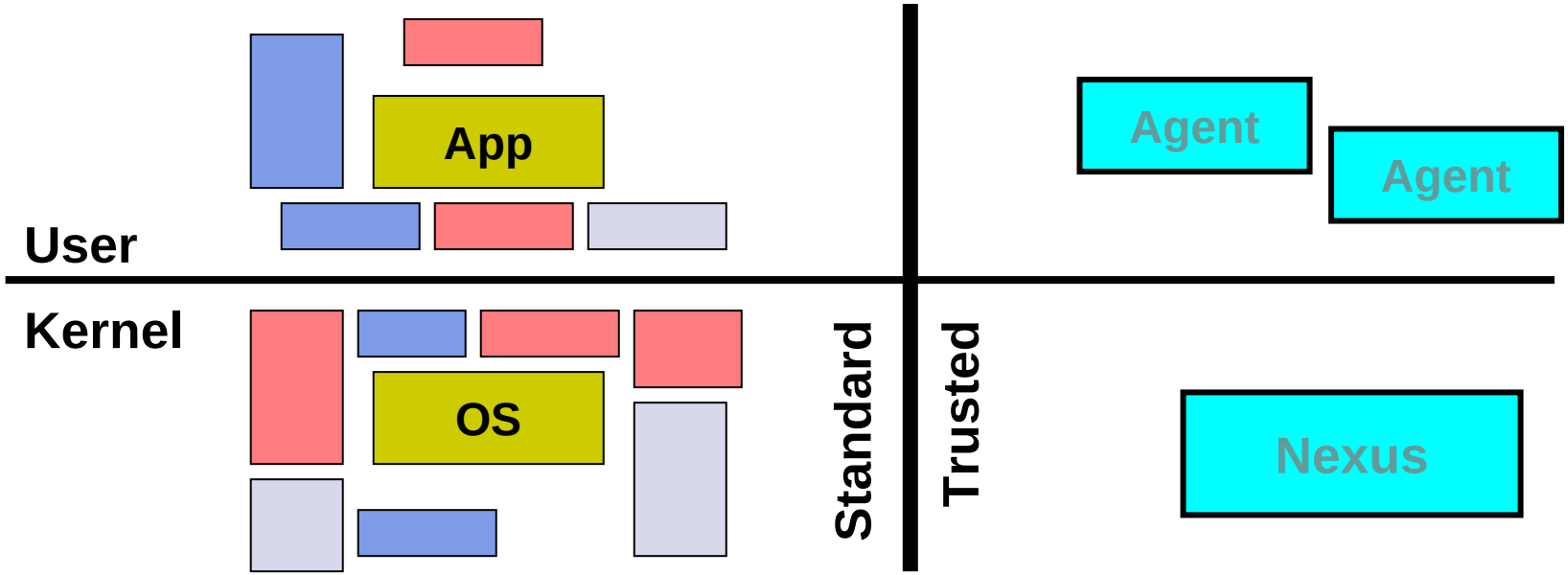
Palladium At 50,000 Feet: 1



- How do you preserve the flexibility and extensibility that contributes so much to the entire PC ecosystem, while still providing end users with a safe place to do important work?
- In particular, how can you keep anything secret, when pluggable kernel components control the machine?

Palladium At 50,000 Feet: 2

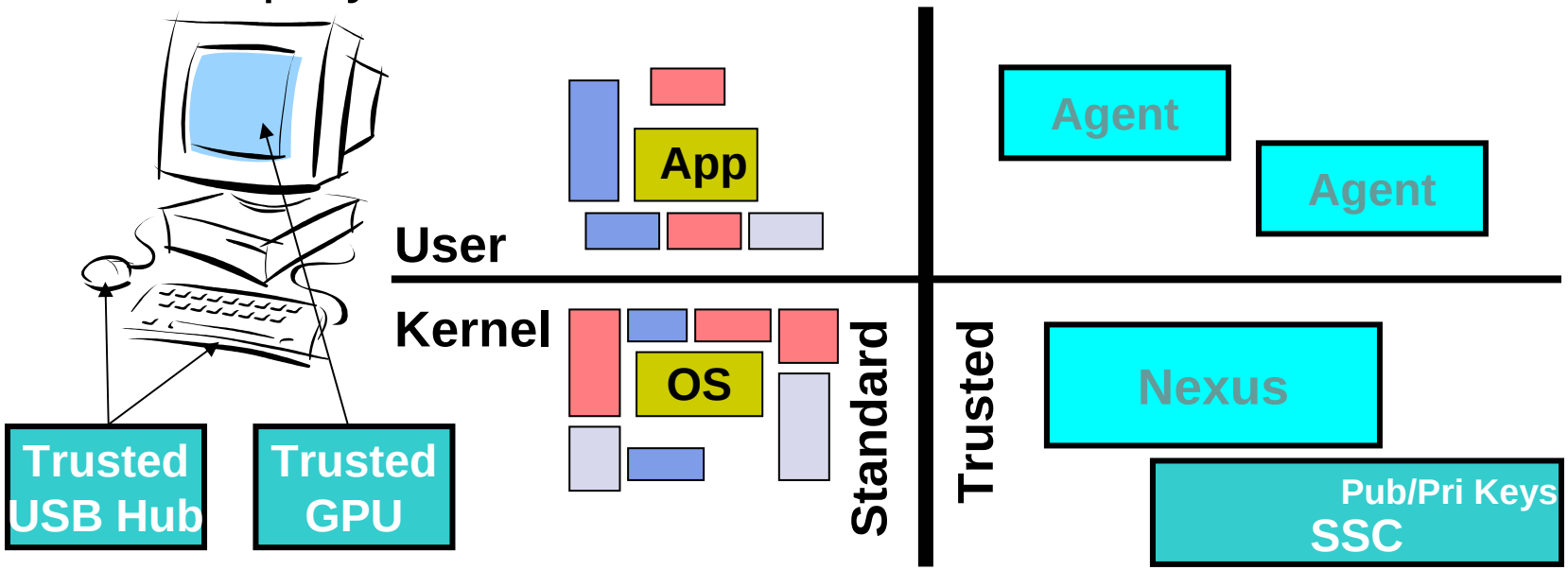
- The solution: subdivide the execution environment by adding a new mode flag to the CPU.



- The CPU is either in “standard” mode or “trusted” mode.
- Pages of physical memory can be marked as “trusted.” Trusted pages can only be accessed when the CPU is in trusted mode.

Palladium At 50,000 Feet: 3

- Agents also need to let the user enter secrets and to display secrets to the user.



- Input is secured by a trusted USB ‘hub’ for KB and mouse that carries on a protected conversation with the nexus.
- Output is secured by a trusted GPU that carries on a crypto-protected conversation with the nexus.
- This gives us “fingertip-to-eyeball” security.

Hardware Summary

- CPU changes
- MMU changes
- Southbridge (LPC bus interface) changes
- Security Support Component (SSC)
  - New chip on the motherboard (LPC bus)
- Trusted USB hub
  - May be on motherboard, in keyboard, or anywhere in between
- Trusted GPU

Hardware Requirements

- SSC – Security Service Component
  - Think “smart-card soldered to the motherboard”
  - Cheap, fixed-function device
  - Contains
    - At least an AES key and an RSA key pair
      - AES key & RSA private key never leave the chip
    - Registers: e.g. the “PCR” (platform configuration register) that contains the digest of the running Nexus
  - Must be close to the chipset (e.g. not a real smartcard) because it must be involved in nexus initialization
  - Contains other security “goodness”
    - RNG, counters, other key-storage, crypto-ops

What Palladium Provides

- Separate protected execution environment for applications (computing agents) that need higher security
  - Hardware-based memory isolation
- Privileged services for these agents
  - Mostly cryptographic services
- Agents can be
  - Standalone
  - Provide services to other applications
- In the long term
  - “Project trust” into the main OS

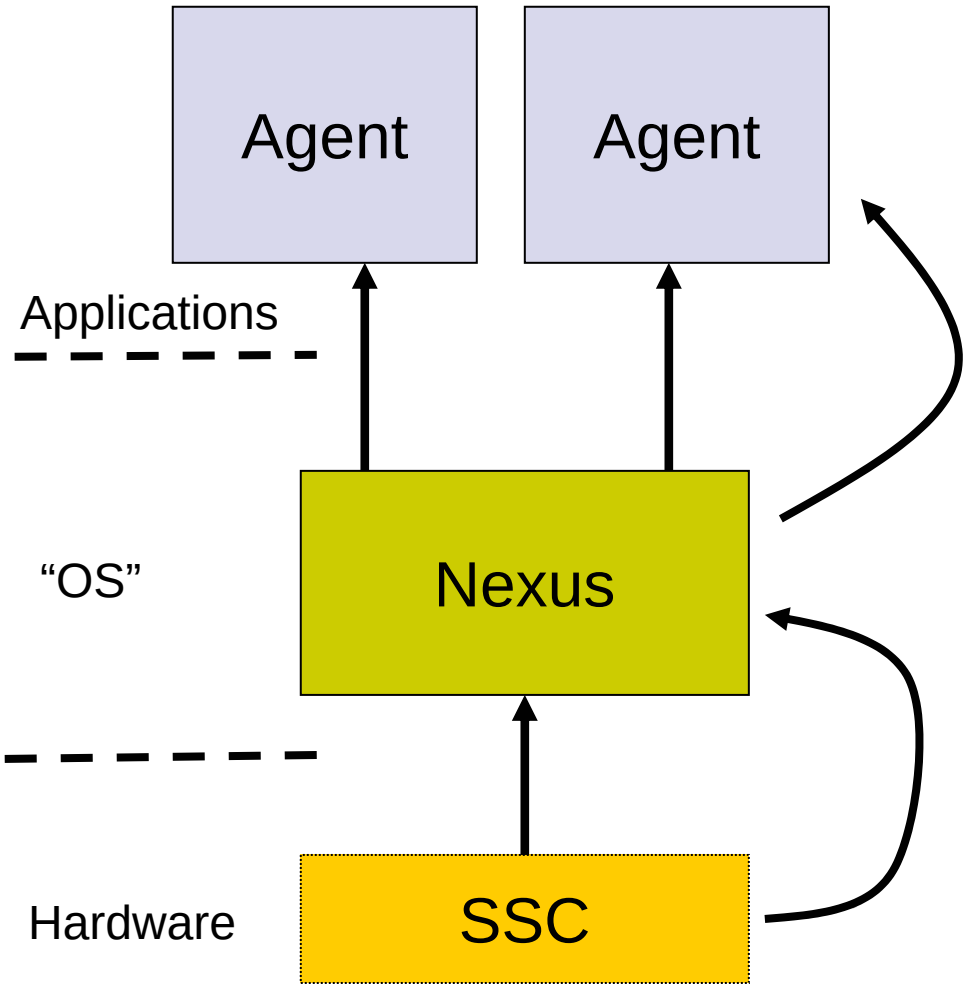
Palladium Core Features

- All Palladium capabilities build off of four key features:
  - Strong process isolation
  - Root key for persistent secret protection
  - Secure path to and from the user
  - Attestation
- The first three are needed to protect against malicious code (viruses, Trojans, etc.)
- Attestation breaks new ground
  - Facts about “things” (SW, users, machines, services) can be proved to (and believed by) remote entities.

Code Identity in Palladium

- The Palladium security model assigns access rights to code identities
  - Palladium always knows what code is running in the right-hand side
- Booting a nexus (security kernel) causes the SSC to compute the hash of the nexus and store it in a read-only register (PCR)
  - Change the nexus, change its identity
- The nexus recursively provides similar features for notarized computing agents executing in trusted mode

Code Identity



App Identity:

- Could be a digest, but we actually use a "manifest" – simplifies management

OS Identity:

- Keep the hardware simple!
- The SSC/chipset measures the digest of the nexus on "secure initialization."

Sealed Storage

- Allows SW to keep long-lived secrets safe from other SW running on the host
  - An encryption technology
    - But more than simple encryption
  - An OS/nexus can keep secrets from other OSs
  - If an OS can keep a secret, it can provide a similar service to applications
- How do we do this?
  - Use the PCR value to “brand” encrypted secrets with the identity of the code that “owns” them.
  - Owners of secrets can also designate alternate recipients (necessary for update & migration)

Sealed Storage
(Allowing code to keep secrets)

- SSC Seal/UnSeal functions
  - Seal(secret, PCR value) -> Blob
    - Says “encrypt this secret so that only the named nexus can retrieve it”
  - UnSeal(Blob) -> secret (or error)
    - If the hash of the current Nexus (current PCR value) is the exact same one included in the blob:
      - Return the secret
    - Otherwise
      - Return an error
  - Implementation: (e.g.) AES using SSC’s key

Attestation

- Attestation lets a remote client know what SW is running
  - OS / Nexus
  - Application
  - Client policy (virus checker, admin access, etc.)
- Attestation is an authentication technology
  - But more than “simple signing”
- Enables authentication of a software configuration (nexus, application process)

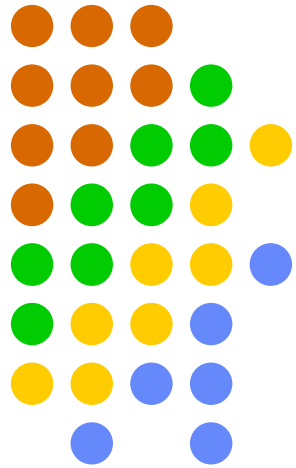
Attestation
(How code authenticates itself)

- SSC Quote Function
  - Quote (string) -> Sign[string | PCR value]
- Protocol building-block:
  - Server/peer:
    - Checks signature
    - Checks certificates on signing key
    - Checks nexus digest is as expected
  - Knows “MS Nexus on Acme Trusted Platform”
- Implementation: RSA using SSC key pair

Secure User Input and Output

- Isolation, sealed storage and attestation aren't enough, however, to keep secrets safe
  - Why?
- Because users can be fooled into thinking they're talking to Palladium when they're not
- We also have to protect the channels to/from the user again sniffing
  - Keyboard, frame buffer, etc.
- User / Application Relationship
  - Protected path between user and application

Policy Issues



Policy Issues

Some of the technical issues we have to solve to make Palladium successful also have policy components to them. For example:

- How do we in practice build an “attestable” TCB?
  - “Attestable” == open, auditable, comprehensible and provable to a remote party
- Since the Pd RSA key pair is unique to the platform, what steps should we take to defend against traffic analysis of user behavior?

Nexus Policies

- Everything that runs today will run on Pd systems
- The platform will run any nexus
  - The user will be in charge of what nexuses he chooses to run
- The MS nexus will run any application
  - The user will be in charge of the applications that he chooses to run
- The MS nexus will interoperate with any network service provider
- The MS nexus source code will be made available for review

Privacy of Machine Identities

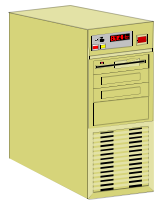
- The issue: Palladium uses at least two sets of unique hardware keys (one AES key, one RSA key pair):
  - These keys are essentially equivalent to unique machine identifiers
  - But this is the only way we can keep your stuff safe!
- Sealed Storage:
  - Uses a unique AES key, but the algorithms are:
    - Opt-in (user designates what software can access the functions)
    - Randomizing (can't decide whether two ciphertexts were created on the same machine)
- Attestation:
  - Uses a unique RSA key, but is designed to authenticate the platform
    - Opt-in (user designates what software can access the functions)
    - We strictly control HW authentication key disclosure
- The hardware has privacy safeguards built into it
  - Access to the RSA public key components is restricted
  - In the current design, only one export of the RSA public key is allowed per power cycle

Pseudo-Identities

- If every party I communicate with needs my HW RSA public key to encrypt some information for me, then that key becomes a platform ID.
  - We need at least another layer of indirection
  - We need to make it easy and cheap to generate temporary pseudo-identities (RSA key pairs) that can be authenticated as belonging to some Pd machine but not any *particular* Pd machine
- Use the HW key once to get the pseudo-identity certified as belonging to a Pd platform, then use the pseudo-identity key
  - Ultimately, this means we need to create a market in pseudo-identities and pseudo-identity providers.

Registering a Pseudo-Identity

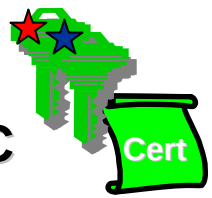
Pseudo-identity certifier/provider (third-party CA)



Send both public keys, proof-of-possession for both keys and the cert to the CA

CA verifies both POPs and the cert, and issues a new cert on the random key that doesn't reference the SSC key or cert

Randomly-generated RSA key pair



SSC key pair and cert from manufacturer

Pd-class PC

Summary

- Palladium is a hardware-based secure execution environment
  - Palladium processes are isolated from each other by the hardware
  - Palladium processes can store & retrieve secrets securely (based on their hash value)
- The nexus provides an execution environment and security/crypto-services to hosted agents
  - Hardware provides crypto services to the nexus
  - Recursively, the nexus provides these same services to agents running on top of it

LCS/CIS Seminar on Palladium

- Want more details on Palladium?
  - Come to my talk tomorrow!
- Friday, 10/18, 10:30am-12pm
- Right here in NE43-518

Questions?