Managing Access Using SSH Keys

Tatu Ylonen
SSH Communications Security
November 8, 2013

USENIX LISA 2013

ssh®
Securing the path to your information assets

SSH – Secure Shell

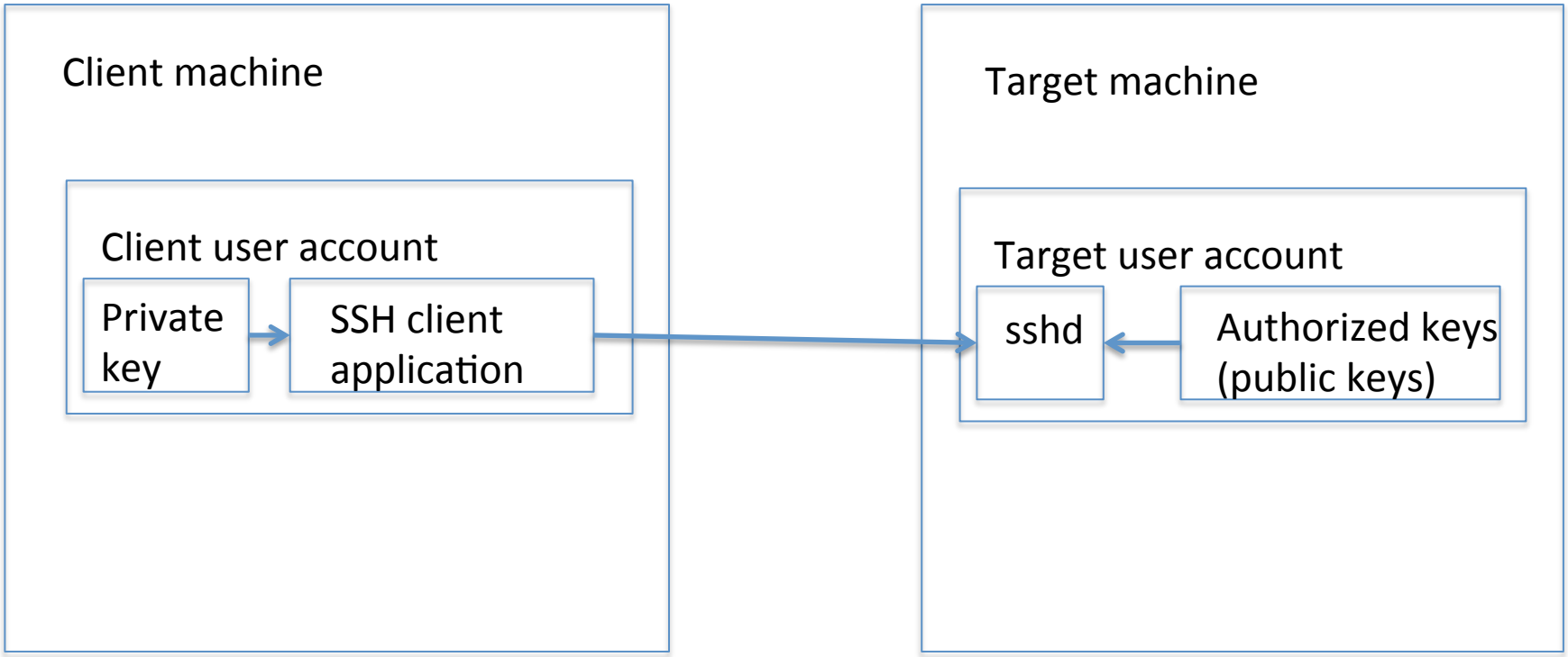
- Tool for remote system administration, file transfers, tunneling – encryption, secure authentication
- Ships with every Unix, Linux, MacOS, most routers, many printers, switches, xDSLs, firewalls; available for every platform (Windows, zOS, tablets, phones) - Uniquitous
- Widely used from automated scripts: systems management, backups, data transfers, patch management, configuration management, disaster recovery systems, cloud provisioning, …
- “Under the hood” in many privileged access management, systems management, network management, version control, backup, and other tools
- OpenSSH, Tectia SSH, and many other implementations

ssh®
Securing the path to your information assets

Public key authentication

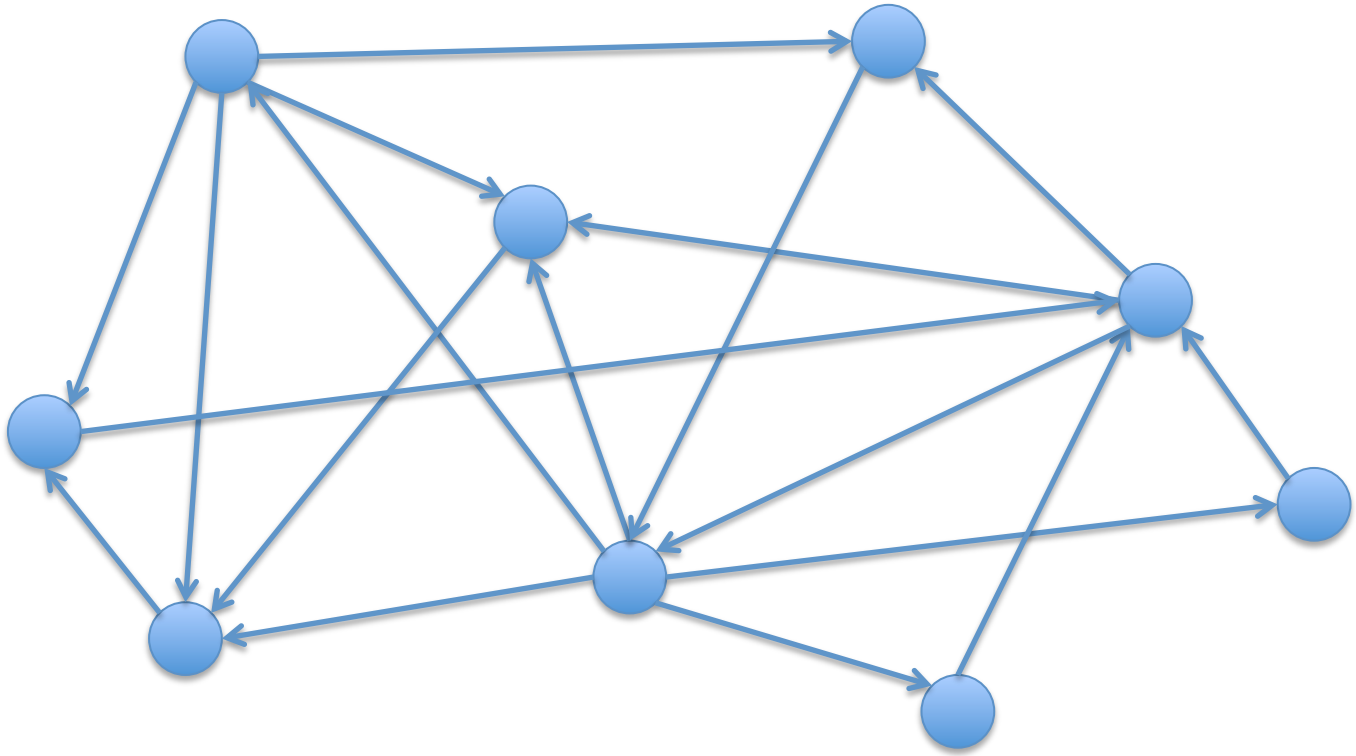
- Digital signature by user's private key used for authentication, verified by authorized public key on server
- The current best practice for configuring automated access between systems
- Basically, create key pair (ssh-keygen), copy public key to user's authorized keys file on server (ssh-copy-id) and you are set
- Possession of private key serves as authentication
- Private keys usually without passphrase when used for automation
- Can configure source restrictions ("from" option), command restrictions ("command" option) but these are rarely used in most environments
- System administrators have been adding authorized keys when convenient for years

Example



- Possession of private key grants access
- Forms trust relationships between systems
- Keys accumulate over time if not managed

So you get



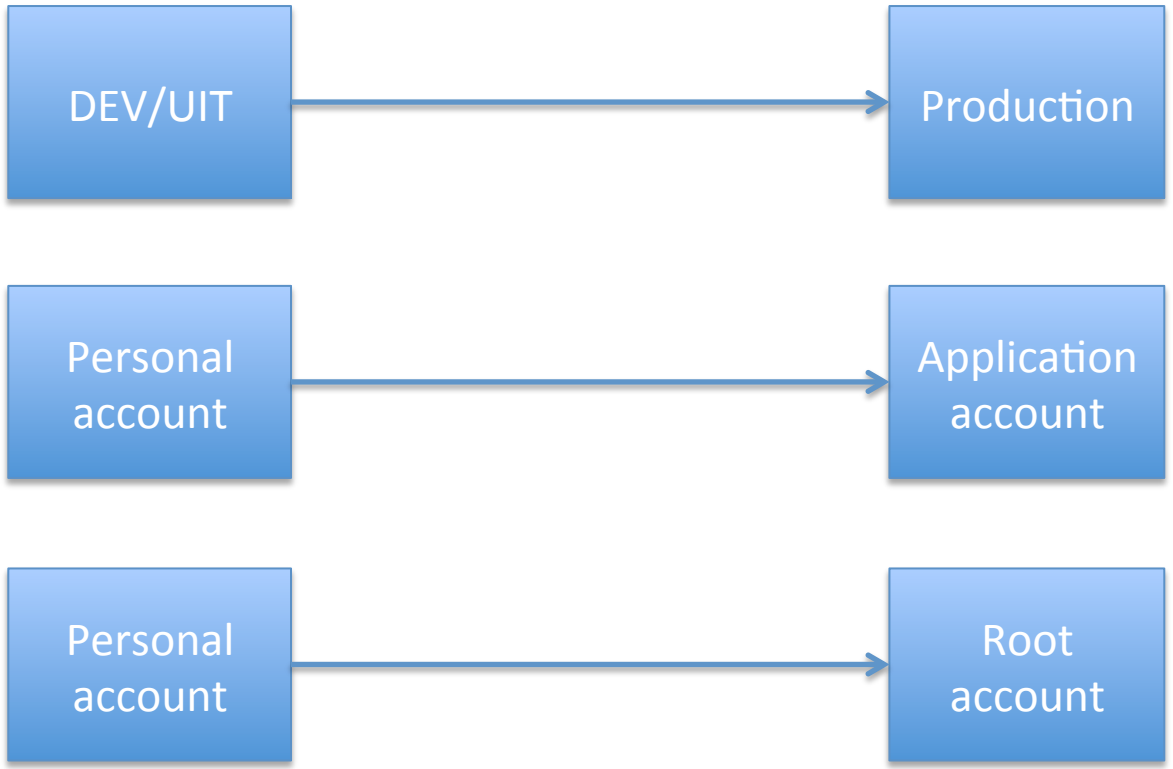
What does your trust graph look like?

ssh
Securing the path to your information assets

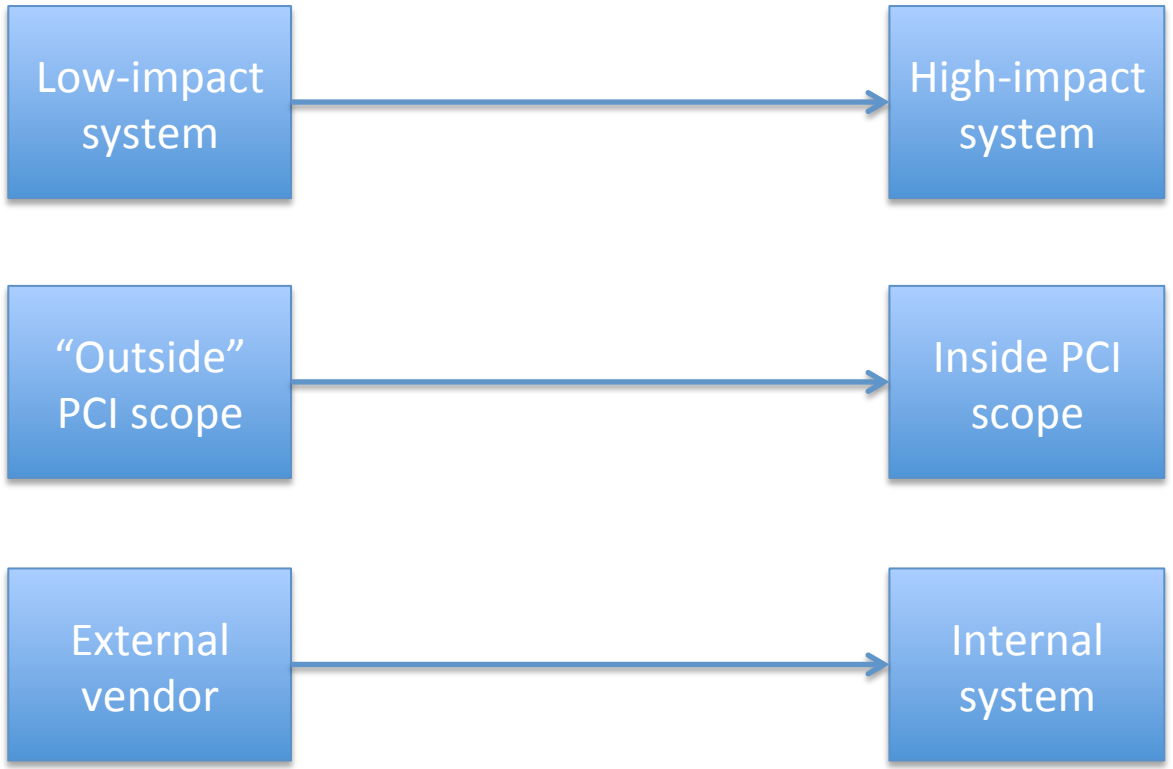
Some real-world examples

- Top-10 global bank
  - Over 1.5 million authorized keys
  - Over 2 million daily key-based logins to just 2000 core production servers (out of 30000)
  - Nobody knew who can access what systems (or is accessing them) using the keys, auditors caught up to it
- "400 000 authorized keys" (top-10 US bank), "probably over 1 million keys" (top-10 US bank), "we were all stunned by the number of keys found" (top-3 credit card company)
- Sometimes >90% of all access credentials to production servers are SSH keys and >90% of OS-level logins to them are using SSH keys
  - CANNOT IGNORE SSH KEYS IN IDENTITY/ACCESS MANAGEMENT and pretend they don't exist

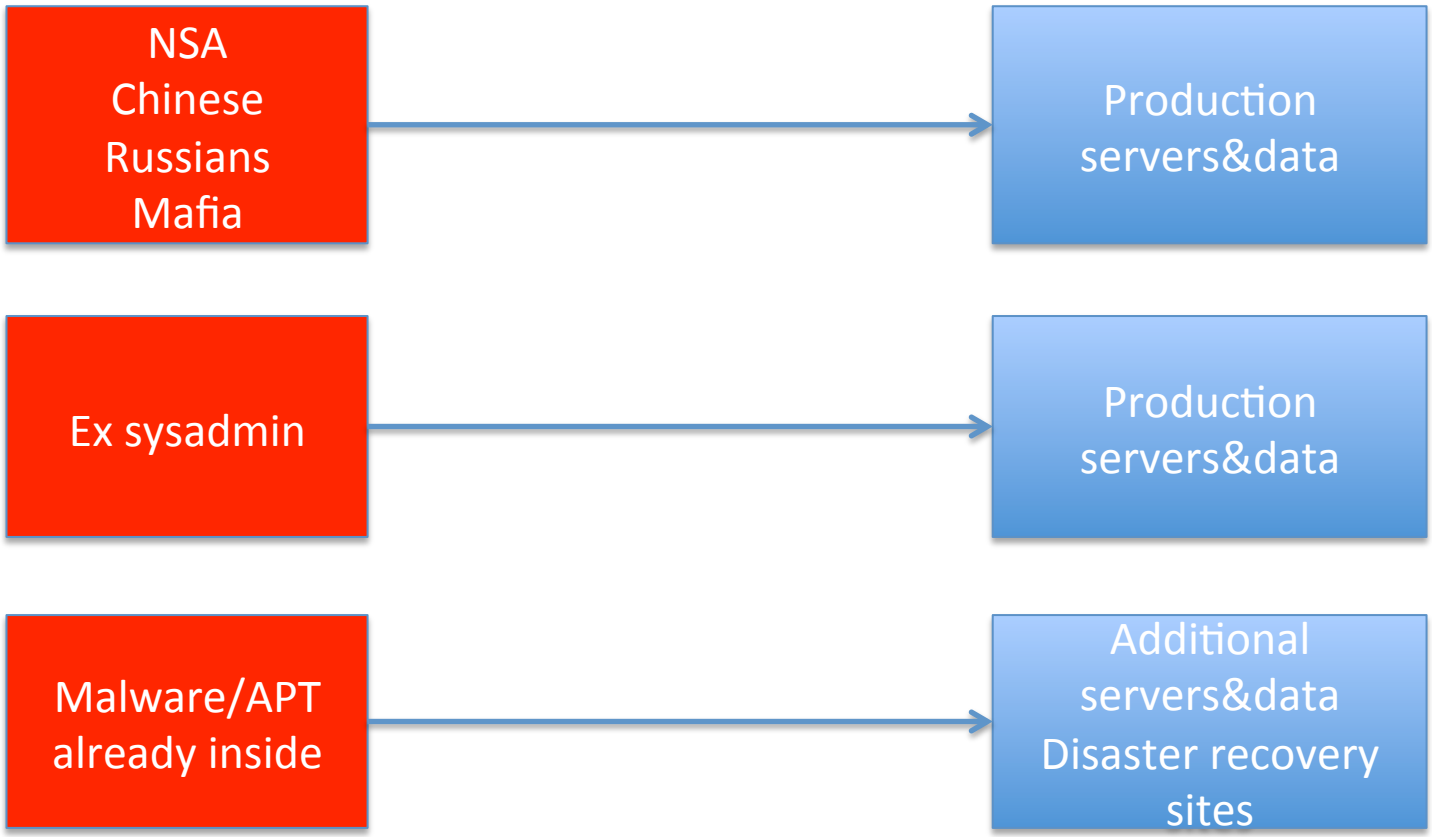
Common trust examples



More common trust examples



Nastier examples



Vulnerabilities arising from mismanaged authorized keys

- Not knowing how can access what
- Not knowing who is accessing what
- Bypassing privileged access management systems
- Backdoors by whomever (what is a better way to hide a backdoor than to add authorized key?)
- Unintended non-file-transfer access by business partner (missing command restriction)
- DEV/UIT->PROD, low impact->high impact, non-PCI->PCI
- Lacking effective termination of access when employee leaves
- Quick attack spread once inside (malware, APT) – SERIOUS!
- Many organizations cannot practically change keys even after breach
- Lack of compliance with PCI, FISMA, Sarbanes-Oxley, FERC/NERC, HIPAA (knowing who can access what; termination of access; auditing privileged access; configuration/security/risk management; PCI scope must include systems having security impact on PCI environment)

Regulatory compliance issues and guidance

- FISMA and NIST SP 800-53 require managing SSH keys in numerous ways (NIST IR coming out very soon!)
- IETF draft-ylonen-sshkeybcp-01.txt provides best practice guidelines (pending update!)
- PCI 3.0 requires addressing SSH key based access in many ways (e.g., scope includes any system having security impact on cardholder data) – whitepapers available at www.ssh.com
- Monetary Authority of Singapore Technology Risk Management Guidelines require managing SSH key based access (e.g., DEV/UIT->PROD) – see whitepaper at www.ssh.com
- FERC/NERC CIP rules require controlling who can configure critical infrastructure
- HIPAA requires controlling who can access patient data
- Sarbanex-Oxley requires controlling who can access financial data
- Most regulatory standards are written technology-agnostic, and don't mention SSH by name – but keys provide system-level access

Roadmap to addressing the issue

- Establish controlled provisioning process
- Continuously monitor and manage SSH credentials for automated and interactive access, support compliance audits
- Remediate existing legacy keys (they cannot be just removed because there can be a huge number of them and many are business-critical!)

Controlled provisioning process

- Enforce approvals for all new key creatio
  - Objective: access provisioning should be on need basis and auditable
- Lock down keys, i.e. move authorized keys to root-owned location
  - Objective: prevent easy bypass of approval process and arbitrary access delegation
- Automate provisioning of approved requests
  - Objective: cost savings, security, less errors
- Document purpose and owner of each authorized key
  - Objective: can't audit and eventually terminate them if purpose not known; need to know whom to ask about key
- Start enforcing command restrictions on keys whenever possible
  - Objective: limit attack spread by malware/APT

Establish continuous monitoring and management

- Both syslog monitoring and key scanning
  - Alert about unauthorized keys and unauthorized key-based access (e.g., copied keys) (objective: know immediately if someone uses a copied or unapproved key)
  - Monitor boundaries (DEV/UIT->PROD, low impact->high impact, personal->service account, external trust, between business units/processes, PCI environment) (objective: compliance, security)
  - Suggest keys that are never used for removal (objective: termination when need for access ceases)
- Facilitate audit of who can access what and that approvals are in place
  - Objective: compliance, know who can access what and why, no unapproved backdoors!
- Implement periodic key rotation
  - Objective: ensures eventual termination by of access by copied keys, particularly important after breach
- Implement privileged access auditing also for key-based access (keys intended for automation may be used by people!)
  - Objective: prevent bypassing privileged access auditing, detect attacks using keys early

Remediating legacy authorized keys

- Problem:
  - There can be lots of them (>1million in some environments)
  - Some are business critical
  - Some are likely backdoors
  - Most organizations have no documentation of them

Remediation approach

- Discover and monitor (objective: understand situation, collect useful data)
  - Find out what authorized keys and identity keys exist
  - Monitor key usage to determine which keys are actually being used and from where
  - Define and monitor policy-violating boundaries (e.g. DEV/UIT->PROD)
- Reduce number of keys (objective: reduce cost of remediation)
  - Eliminate keys that are never used
  - Eliminate policy-violating/boundary-violating keys (DEV/UIT->PROD, personal->service etc)
- Identify business purpose and owner for each remaining keys (objective: grant access on need basis only, eliminate backdoors, achieve compliance)
  - Collect approvals from application teams
  - Record purpose, owner, approval in database for audit and maintenance
  - Eliminate keys for which no business purpose can be found
- Rotate all keys (objective: make copied/leaked keys unusable)
- Add command restrictions to all remaining keys if possible (objective: prevent malware/APT attack spread)
- Optionally add source restrictions to all keys (objective: make use of copied keys harder)
- Must always have a back-off option if some key turns out to be important!

Some experience

- We have now worked on remediation projects with some customers for over an year (e.g., 10 people onsite at a bank, two of them from Deloitte), using Universal SSH Key Manager tools
- The scope and complexity of the problem has surprised us, but project is on track
- Biggest cost is remediation, particularly getting signoff from application teams for the keys
- Remediation of a million keys is multi-year project
- Planning of the remediation project, tool choices, and related policy decisions have great impact on total project size, cost, and duration
- The approach presented here represents current best understanding of what needs to be done
- Sometimes people think PKI or AD/Kerberos are silver bullets, but there is still the legacy infrastructure, old trust relationships (many business critical!), and you will run into same issues with Kerberos keytabs (+ OpenSSH by default implements SSO -> many implicit trust relationships, and does not support command restrictions for Kerberos access -> attack spread cannot be limited)
  - That said, PKI and AD/Kerberos can still be useful for other purposes

Software tools

- Software packages for SSH key management are available from several vendors
  - Some only do discovery, some claim more than they can deliver
- Most of the cost is in remediation, so any assistance tools can provide for remediation is critically important
- Ongoing provisioning and cost savings in provisioning are going to be a big factor long-term
- Solutions from SSH Communications Security:
  - Universal SSH Key Manager – SSH key/access management, the only SSH key/access management solution that has seen substantial deployment in the real world so far; supports OpenSSH, Tectia SSH, and several other SSH implementations
  - CryptoAuditor – privileged access auditing/file transfer auditing and control for SSH/SSH/RDP (transparent, works also for automated key-based access, sysadmins can continue using their preferred client-side tools!)
  - Consulting and remediation services are also available.

Conclusion

- Most organizations with large Unix/Linux environments have a serious security and compliance problem with unmanaged authorized keys
- The scope and impact of the issue is not yet very widely understood
- Knowing who can access what systems and information is critical for information security – without it you don't have confidentiality, integrity or continuity
- It is about access, not cryptographic algorithms or sizes
- For more information:
  - http://tools.ietf.org/id/draft-ylonen-sshkeybcp-01.txt