



IBM Research

Trusted Linux Client

Dave Safford, IBM Research

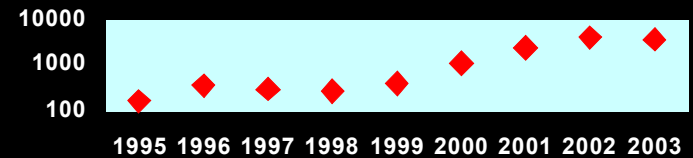
Trusted Linux Client (TLC): Outline

- Problem
- Goals
- Status
- Demo
- Next Steps
- Architecture

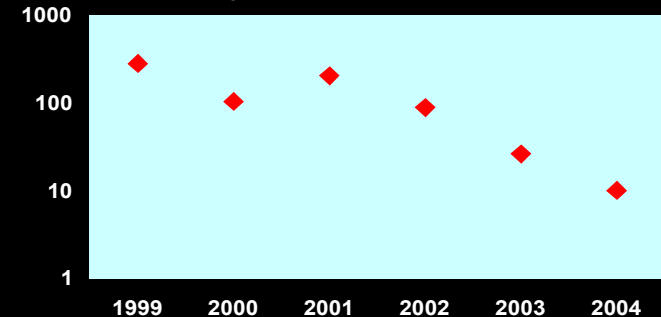
The Problem: Client Risk is Dramatically Rising

- **The number of attacks in the wild, and their lifetimes and impact are growing fast**
 - 450% increase in Windows viruses over last year
 - 1500% growth in BotNets Jan to Jun 2004
 - The myDoom.O virus overloaded networks around the world in August 2004
 - Blaster worm attack cause First Energy's Davis Besse Nuclear Reactor to loose digital control for over four hours in January 2003
 - Viruses are already deploying attacks against AV software
 - 80% of clients have spyware infestations
 - 30% of clients already have back doors (FSTC)
- **Increase in vulnerability rate is slowing, but the time between the publication of a security vulnerability and the broad exploitation of it is markedly decreasing**
- **Financial losses rapidly increasing:**
 - Phishing attacks: \$500M direct losses in first half of 2004
 - Identity theft is the fastest growing crime in US

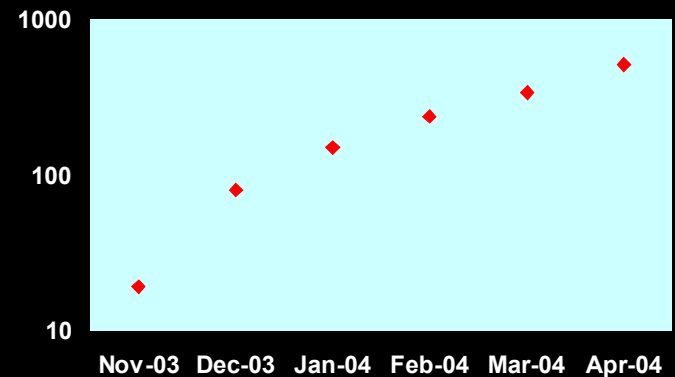
Discovery of Vulnerabilities *



Days to Broad Exploitation **



Unique Kinds of Phishing Attacks **



* cert.org Nov 2004

** July 2004 Information Security

What does TPM do?

- RSA crypto
 - key generation, signature, encrypt, decrypt
- Secure storage
 - private keys
 - master keys (eg loopback)
- Integrity measurement
 - Platform Configuration Registers (PCR)
 - compromise detection
 - Tie key use to uncompromised environment
- Attestation
 - host based integrity/membership reporting



Understanding TCPA:

- Main Specification:
 - Trusted Computing Platform Alliance (TCPA)
home page and main specification v1.1b:
<http://www.trustedcomputing.org>
 - Trusted Computing Group (TCG) home page:
<http://www.trustedcomputinggroup.org>

- Problem:
 - Spec is over 320 pages
 - very hard to understand

Help in Understanding TCPA:

- Tutorial/Introduction paper: (4 pages)
Linux Journal, August 2003
- White papers, open source code
<http://www.research.ibm.com/gsal/tcpa>
device driver/access library/example applications
- Bottom line:
Open source let's you
see for yourself
better understand the chip

Programming view of the TPM

Functional Units	Non-volatile memory	Volatile memory
RNG	Endorsement Key (2048b)	RSA Key Slot-0 ...
Hash	Storage Root Key (2048b)	RSA Key Slot-9
HMAC	Owner Auth Secret (160b)	PCR-0 ...
RSA Key Generation		PCR-15
RSA Encrypt/Decrypt		Key Handles
		Auth Session Handles

Trusted Linux Client Goals:

- Take advantage of new security technologies:
 - PCD support and availability of Trusted Platform Module (TPM)
 - Linux Kernel 2.6.0 with the new Linux Security Module (LSM)
- Leverage Existing IBM Research Projects
 - TPM open source drivers, library, and utilities
 - TPM Key backup/restore/migration server
 - TPM key PKI/LDAP integration
- Create new kernel security modules:
 - boot time integrity measurement
 - run time integrity measurement for all files
 - integrity enforcement with mandatory access controls

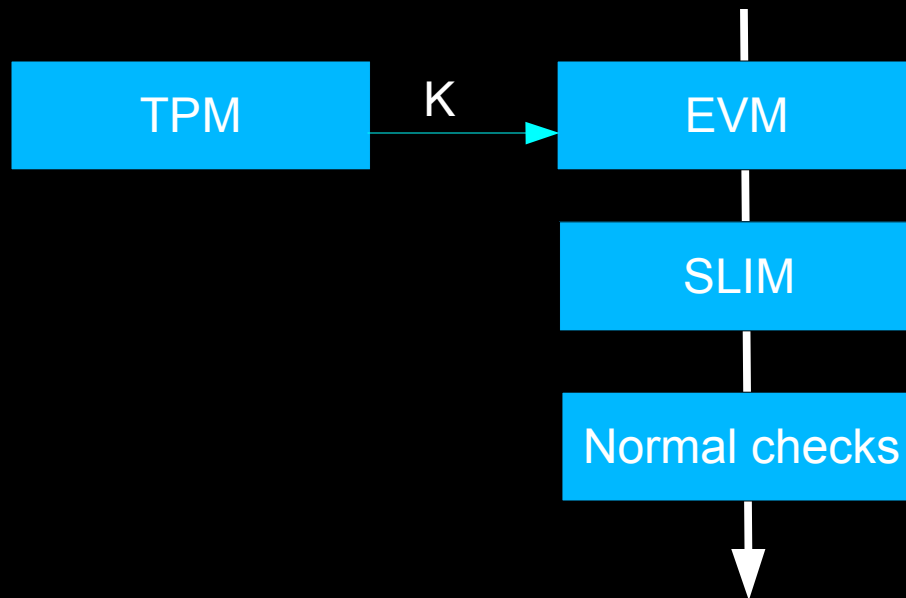


Integrated Client Security Technologies

- T42 Boot time:
 - Integrated fingerprint sensor supplies:
 - Power On Password (boot not allowed without password)
 - Hard Disk Password (hard disk locked without password)
- Trusted Linux Client boot time:
 - TPM measures kernel and initrd
 - if kernel and initrd good, TPM unseals kernel master key
- Trusted Linux Client run time:
 - all security xattr are HMAC'ed with kernel master key
 - every file is checked against the authenticated hash
 - mandatory access control rules protect system integrity

Trusted Linux Client LSM Modules:

- TPM: driver measures integrity of kernel and initrd, and releases kernel key
- EVM: Extended Verification Module – authenticates extended attributes, data
- SLIM: Simple Linux Integrity Module – Mandatory Access Control Sandbox
- Implemented as stacked LSM module:



TPM Module

- TPM measures integrity of boot process through kernel and initrd
- In initrd boot, user supplies sealed kernel key and authorization PW
- If TPM measurements match, and password matches, TPM releases K.
- Master key K is used to generate derived keys for
 - encrypted home directory partition loopback
 - authenticated file attribute checking (EVM)

Extended Verification Module: EVM

- Use extended file attributes to store authenticated file metadata
 - file hash
 - mandatory access control labels
 - version
 - antivirus status
- Use tpm based symmetric kernel key to HMAC these attributes
- Verify file once at open/execute, and cache verification
- “heavy lifting” done at install time, runtime is just file hash and HMAC
- Extensible, policy based definition of attributes and actions

SLIM: Mandatory Access Control Alternatives

- Bell and LaPadula:
 - Secrecy based Mandatory Access Control
- Biba:
 - Integrity based Mandatory Access Control
 - Low/High-watermark option
- Lomac: Linux module version of Biba Low-watermark
 - Only 2 integrity levels *trusted* & *untrusted* make it easy to administer
 - All process start out trusted, only when a process attempts an untrusted operation is it demoted (i.e. reading from the network)
 - Problem: guard processes to promote untrusted objects to trusted.
- Caernarvon: modified Biba integrity model
 - Incorporated support for guard processes, verified trusted programs, which are allowed to remain high integrity, while reading low integrity objects
 - Separation of reading and writing/execution permissions
 - Does not support low/high-watermark

SLIM: Mandatory Access Control Alternatives Cont.

- Security Enhanced Linux (SELinux): LSM framework based
 - Permits the creation of a wide range of security models
 - Default security model: Type Enforcement (TE)
 - More powerful than Biba or Caernarvon
 - Problems:
 - Difficult to configure: default configuration file ~33,000 line ruleset
 - Default configuration did not ensure integrity containment
 - Cannot create low/high watermark policies

SLIM Sandbox:

- **Simple Linux Integrity Module (SLIM)**
 - Use of LSM framework hooks
 - EVM context information to enable sandbox decision
 - Includes Lomac's low-water mark integrity model for ease of administration
 - With Caernarvon's separation of read and write/execute permissions
 - With Caernarvon's signed guard processes – verified trusted programs
- **Basic Integrity Operation:**
 - Low Integrity processes can read and execute up, but not write up
 - High Integrity processes can write down, but are demoted on read/execute down
 - Trusted “guard” processes, verified by EVM, can read down without demotion
 - rpm
 - sshd

SLIM Access Classes

All Files are labeled with an Integrity and Secrecy MAC label

Integrity Access Classes (IAC)

SYSTEM

USER

UNTRUSTED

EXEMPT

Secrecy Access Classes (SAC)

SENSITIVE

USER

PUBLIC

EXEMPT

All Processes have upper and lower Integrity and Secrecy labels:

Integrity Write/Execute Access Class (IWXAC)

Integrity Read Access Class (IRAC)

Secrecy Write Access Class (SWAC)

Secrecy Read/Execute Access Class (SRXAC)

(Upper and Lower are the same, except for guard processes.)

EVM and SLIM Extended Attributes

EVM Extended Attributes:

```
security.evm.hash - hash of file data (from signed rpm)
security.evm.hmac - hmac-shal of security.* attributes
security.evm.packager - signer of package
security.evm.version - version of package
```

SLIM Extended Attributes

```
security.slim.level - six class values (values are space delimited)
```

```
IAC - File's Integrity Access Class
```

```
SAC - File's Secrecy Access Class
```

```
IRAC - guard process Integrity Read Access Class
```

```
IWXAC - guard process Integrity Write/Execute Class
```

```
SWAC - guard process Write Access Class
```

```
SRXAC - guard process Read/Execute Class
```

EVM and SLIM Extended Attributes

Attributes of `"/bin/ping"`:

```
[root@localhost safford]# getfattr -d -m "^security" /bin/ping
security.evm.hash="\1\265Ad6a4d94fb694cffd2847acf40dbc6485"
security.evm.hmac=0s8ZT9309FkQBag7HkqqieSeuya3s=
security.evm.packager="\1\265ARed Hat, Inc.
    <http://bugzilla.redhat.com/bugzilla>"
security.evm.version="\1\265A20020927(rel16)"
security.selinux="system_u:object_r:bin_t\000"
security.slim.level="SYSTEM"
```

Running `"/bin/ping"` first time (from `dmesg` logging):

```
evm_calc_hmac: ping - security.evm.hash included
evm_verify_xattr: verification of security.evm.hmac succeeded
evm_inode_permission: 'ping' HMAC verification succeeded
evm_inode_permission: 'ping' HMAC verify xattr
evm_inode_permission: security.evm.hash is d6a4d94fb694cffd2847acf40dbc6485
evm_inode_permission: security.evm.hash succeeded
```

Running `"/bin/ping"` subsequent times:

```
evm_analyze_cacheinfo success
```

TLC: the Future

- Possible TLC follow-on enhancements
 - integration of anti-virus checking with optimum performance
 - integration of patch management

- The bigger picture:

We need a more general solution

linux and windows

cross platform, open architecture

strong defenses, and rapid recovery

leverage linux, TLC for part of the solution

use OpenHype for integration/isolation

Hypervisor technology provides strong isolation and controlled sharing among applications

