Protegiendo sus ingresos, con esquemas de licenciamiento aplicadas al desarrollo de Software

Víctor Montes de Oca

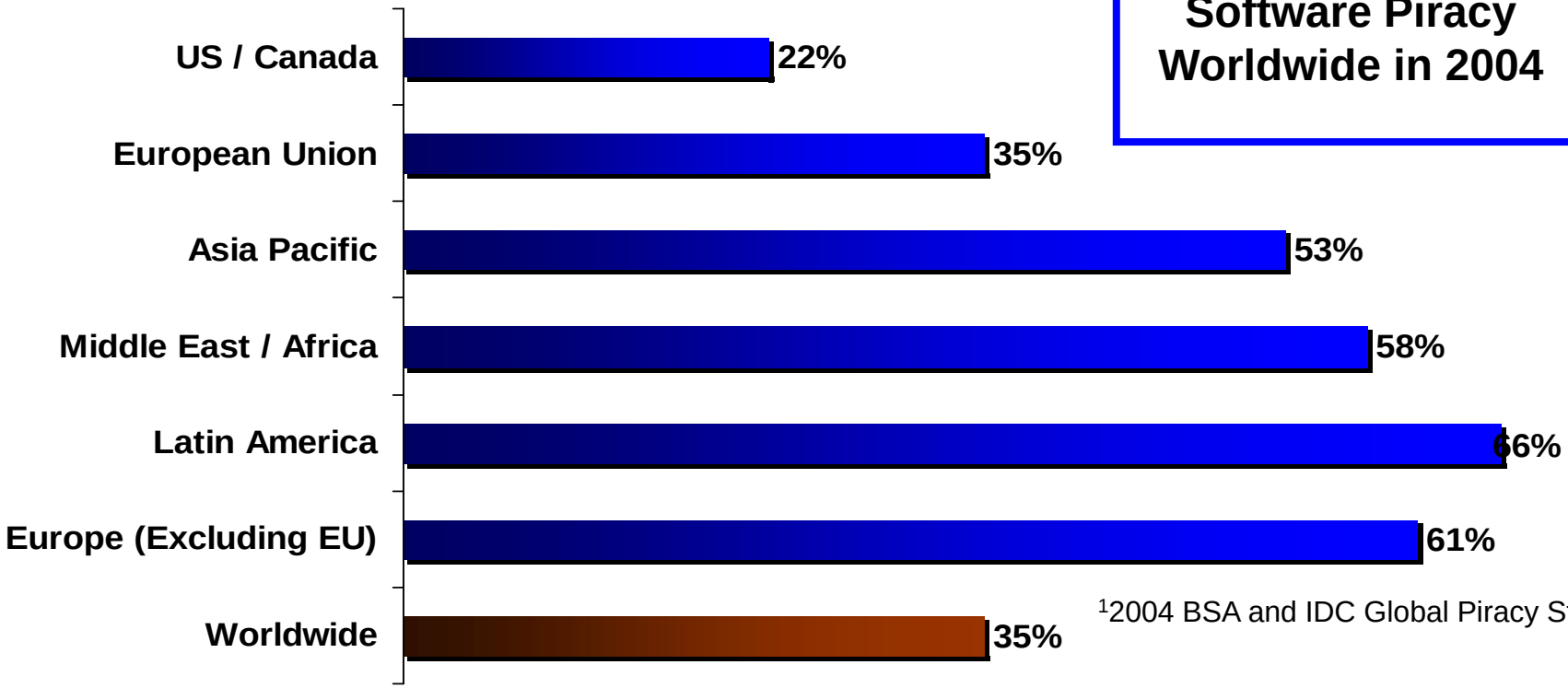
Director Técnico, SafeNet América Latina

Agenda

- Software Piracy
- Who Pirates & Why?
- Anti-Piracy Solutions
- Traditional Implementations and Hacks
- Implementing the Latest Technologies
- Other Considerations
- Summary and Questions

Software Piracy Problem

Piracy Rate by Region[1]



Over $33B Lost to Software Piracy Worldwide in 2004

[1]2004 BSA and IDC Global Piracy Study

Down from 36% in 2003 but losses increased by $4 billion

Software Piracy Problem – Revenue

Example:

- XYZ Corp. sells application ABC for $2,500/seat
- Expected number of applications sold/year – 2,000
- Expected revenue from sales – $5,000,000
- Piracy rate in the US – 22%
- Revenue loss to piracy - $1,100,000

If the XYZ Corporation sells its application outside the US, losses could be much higher

Who Pirates/Why?

- Casual Copiers
  - Buy one copy and copy to other machines
  - Give to friends for copying
- Crackers
  - Do it for the thrill of the challenge
  - Notoriety in the “community”
- Hackers
  - Hack software for a profit

High speed internet connections make it easy to download pirated software

Anti-Piracy Solutions

- Software Based
  - Locked to a node on a user's computer
  - Typically low-cost high volume software
  - License management
  - Keeping the honest people honest
  - Lower security than hardware

Anti-Piracy Solutions

- Hardware Based
  - Physical security – universal serial bus (USB) or parallel port
  - License is stored in a key for security and portability
  - License models can be incorporated
    - Entire application
    - Modules
    - Features
  - High value software applications
    - Applications that are susceptible to hacks
  - End-user considerations

Traditional Implementations

- APIs
  - Query tables - calls are made to check the presence of a key
  - Code obfuscation to protect against reverse engineering
  - Shared secret challenge response
- Wrappers
  - Protect .exe's and dll's
  - Quick time to market
  - Low level of application security

Common Mistakes and Vulnerabilities

- Small query tables
- Only few calls made to the key
- Application only checks for presence of a license
- Shared secrets are stored in the software application and can be hacked
- Wrappers
  - Used as only level of protection
  - Generic implementation – not unique

Types of Hacking Techniques

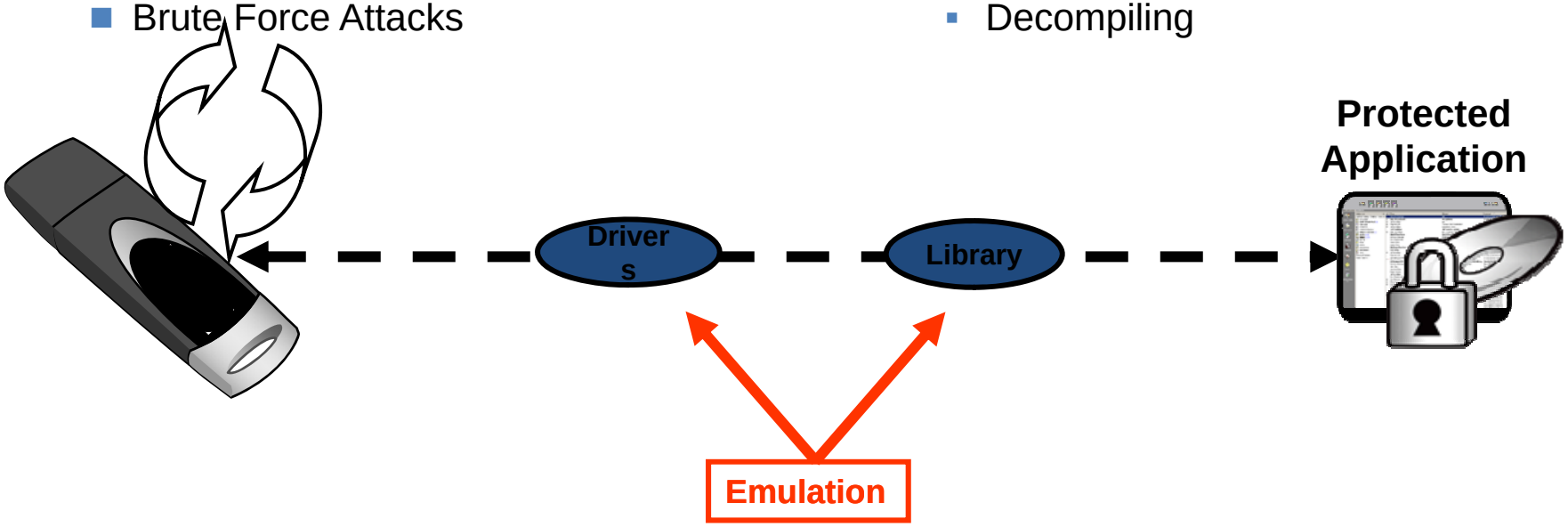
- Record and Playback
- Driver and Library Emulation
- Brute Force Attacks
- Reverse Engineering
  - Debugging
  - Decompiling

Cracker community savvy and increased computing power requires constant security innovations

Points of Attack

- Record and Playback
- Reverse Engineering
  - Debugging
  - Decompiling

- Brute Force Attacks



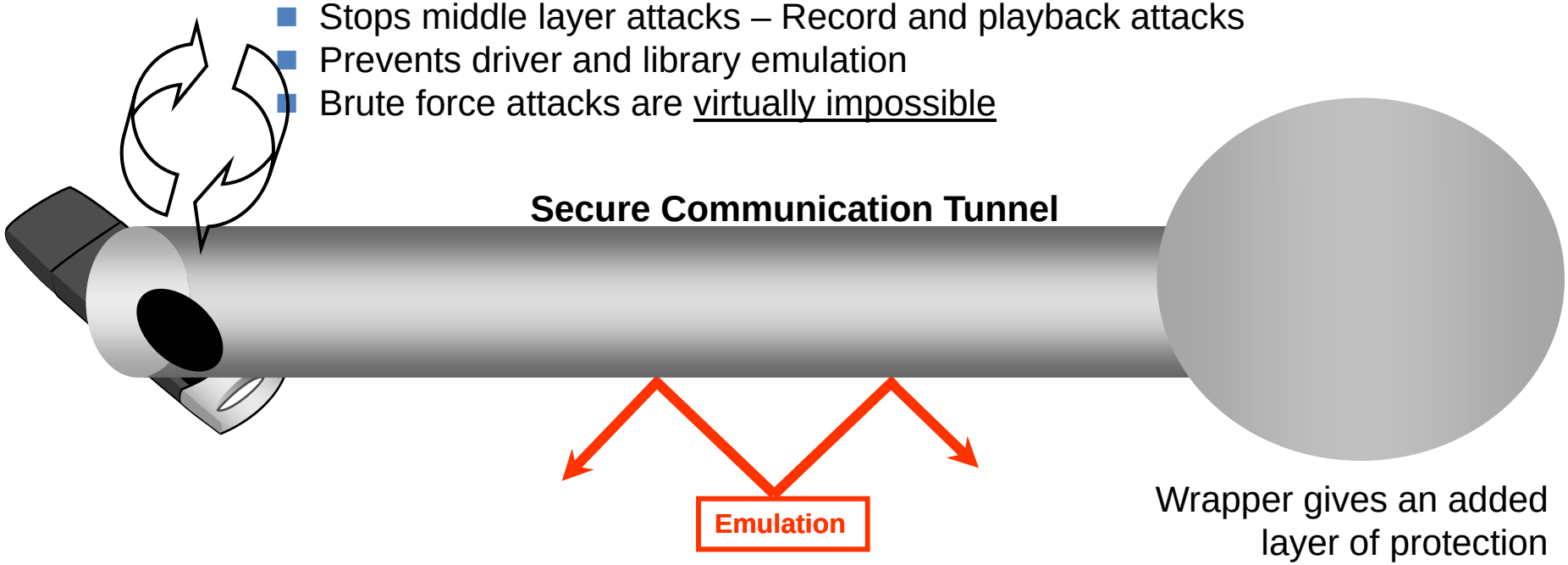
Cracker community savvy and increased computing power requires constant security innovations

Implementing the Latest Technologies

Public Key Cryptography

- Secures communications between application and key
- Public key cryptography generates keys for AES encryption
- Secret is never revealed

- Brute Force Attacks
  - Stops middle layer attacks – Record and playback attacks
  - Prevents driver and library emulation
  - Brute force attacks are virtually impossible



Wrapper gives an added layer of protection

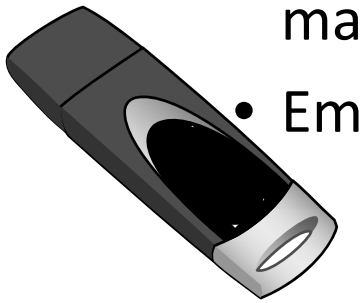
Implementing the Latest Technologies

- **Wrappers**
  - Multi-layer wrapper
  - Debugger protection
  - Code Compression
  - Protection against decompiling
  - Security Enhancement

Implementing the Latest Technologies

Internal Authentication

- Internal checks to ensure that firmware is bound to token
- Prevents cloning of the keys at every level of the manufacturing process
- Emulation/Cloning of a key is virtually impossible



Other Considerations…

- Ease and Speed of Implementation
- License Models
  - Trial/Demo
  - Subscriptions
  - Pay Per Use
- Secure Distribution Channels
- Securely Update Field Deployments

Ease of Use –
High Level API Implementation For Demo License

Low-level APIs

Complex Multi-Step Process

- Determine memory to allocate for dates
- Create Conversion Formula (Date in a Hex Format)
- Store dates in memory and determine calls
- Programmatically enforce demo period and time tampering

High-Level APIs

- Developer enters date in dd/mm/yy Format.
- **DONE**

High Level APIs make the calls to low level APIs, automatically converting, adding security elements, and storing the routine in memory

Other Considerations…

- Ease and Speed of Implementation
- License Models
  - Trial/Demo
  - Subscriptions
  - Pay Per Use
- Secure Distribution Channels
- Securely Update Field Deployments

Summary

- Revenue loss due to piracy continues to increase
- Broadband making the cracking community much more effective and downloading easier
- Advancements in hacker technologies require periodic anti-piracy updates
- Public key cryptography, strong encryption and internal authentication offer the most secure implementation
- Ease and speed of implementation should be considered:
  - Reduction in development costs
  - Time to market

¿ Preguntas ?

Muchas gracias !