VICTIMS OF FRIENDLY FIRE
(It’s kinda ugly out there…)

virus BULLETIN

2010 CONFERENCE
Vancouver, 29 September 2010

Speaker: C. Ronchi, EISST Ltd

EISST Ltd
Fairfax House,
15 Fulwood Place
London WC1V 6AY, UK
T: +44 (0)20 79 695 688
F: +44 (0)20 77 483 273
E: info@eisst.com
W: www.eisst.com

Is it *really* that bad?

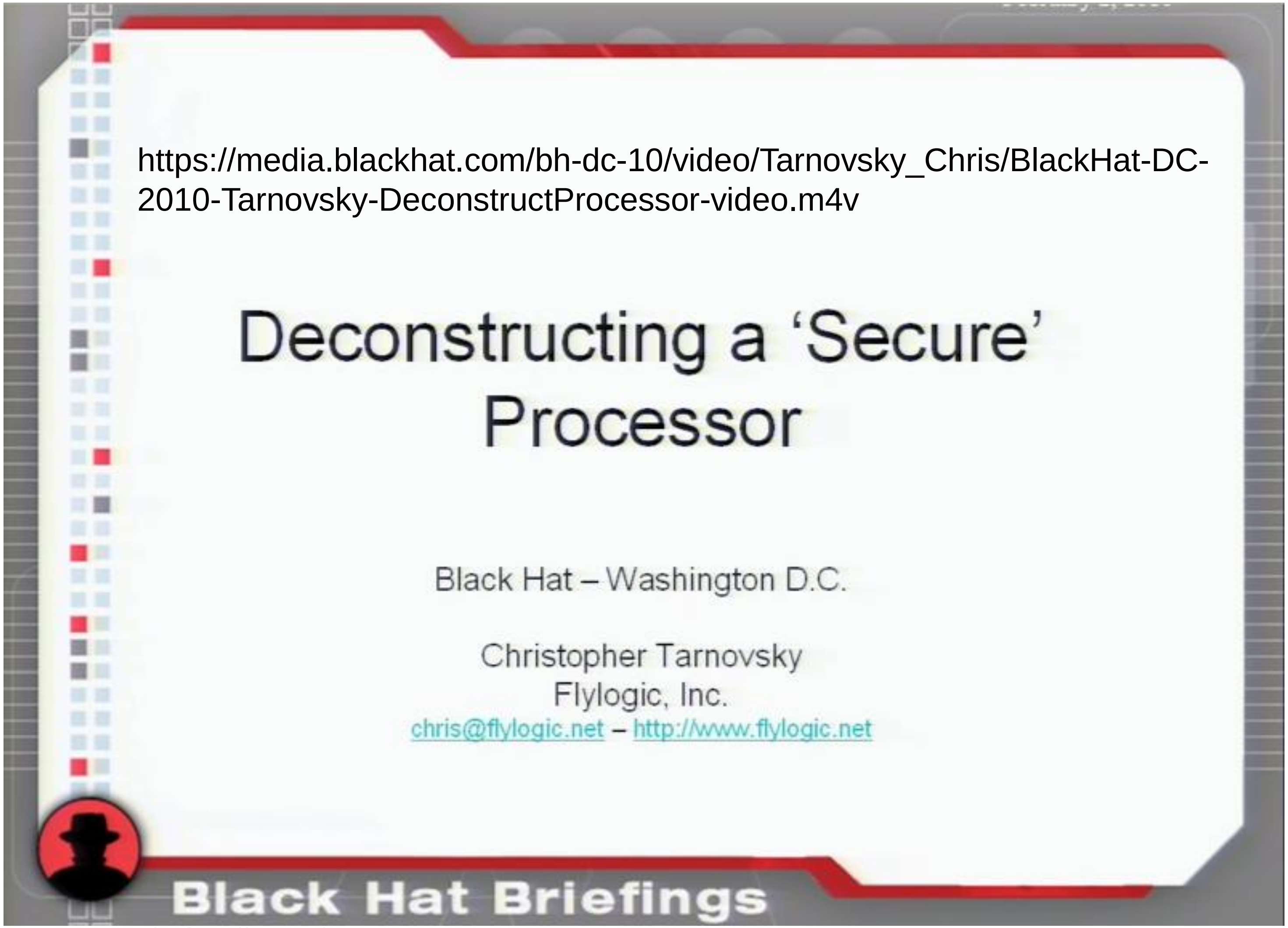
Fact 1

All components enabling a typical online transaction can be hacked.

Smart Card: 100k$
Browser: 1k$
PC : 100$
User: 1$

1. Facing Reality: News from the Battle Ground

The Infineon SLE66CX642 (64K) and the SLE66CLPE were both successfully attacked and read !



https://media.blackhat.com/bh-dc-10/video/Tarnovsky_Chris/BlackHat-DC-2010-Tarnovsky-DeconstructProcessor-video.m4v

Deconstructing a ‘Secure’ Processor

Black Hat – Washington D.C.

Christopher Tarnovsky
Flylogic, Inc.
chris@flylogic.net – http://www.flylogic.net

Black Hat Briefings

Is it *really* that bad?

Fact 2

There is no (provably secure**) system that can withstand the test of fire in real-life online transaction scenarios.**

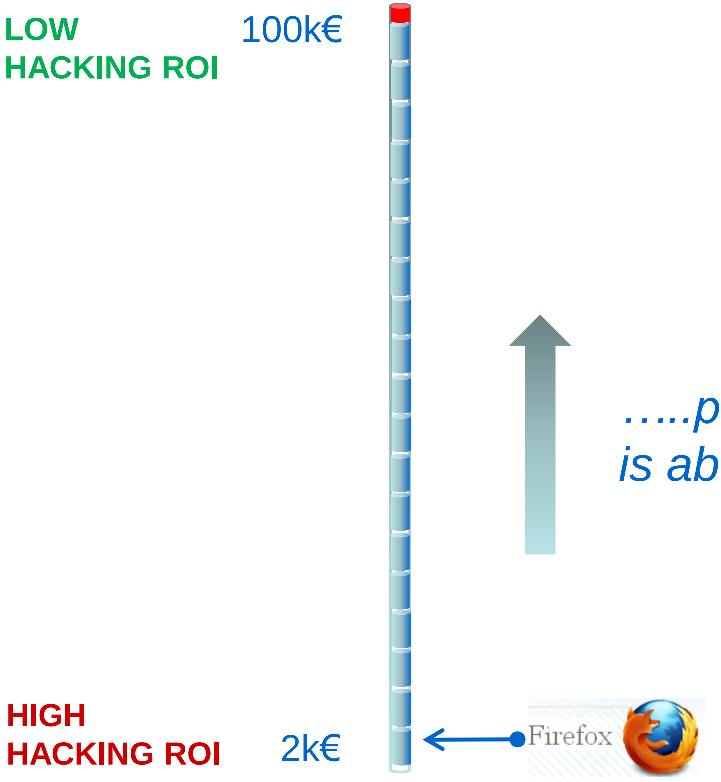
Social engineering (supported by the natural tendency of Humans to trust others**) is one hard limit to the maximum level of security attainable.**

Is it *really* that bad?

Fact 3

Practical security (which is the only concept relevant to users**) is ultimately about the *economy* of hacking, not about the *technology* of vulnerabilities.**

MEASURING SECURITY BY HACKING ROI



…..preventing online crime is about ***economics!***

Firefox with OTP or SMART CARD Reader

Is it *really* that bad?

Fact 4

Institutions do not care about attaining higher security at the cost of much lower transaction *efficiency*.

End Users do not care about attaining more security at the cost of less *usability*.

Is it *really* that bad?

Fact 5

Anti-Virus and Anti-Malware are prevasive, yet cannot fully withstand the variety, volume and sophistication of attacks from hackers supported by the e-crime industry.

So the question arises as to how to handle the cases when (not *if* **) attacks are successful.**

Malware's Attack & Defense Techniques

Armoring:

A mechanism employed by malware for the purpose of impeding its analysis. Such mechanisms are typically targeted at specific analysis methods (e.g. binary packing, anti-debugging mechanisms, virtual machine checks, dynamic analysis)

Infection/Persistence:

A process by which malware instantiates or "installs" itself on a system and ensures continual execution (e.g. injecting a malicious binary into a process address space, executing a malicious process, insertion of an auto-run registry key, installation of a malicious binary as a system service)

Metamorphism/Polymorphism:

A mechanism used by malware for automatically re-coding itself each time it propagates or is otherwise distributed, and for changing the appearance of its run-time code, primarily for the purpose of evading detection through physical signatures.

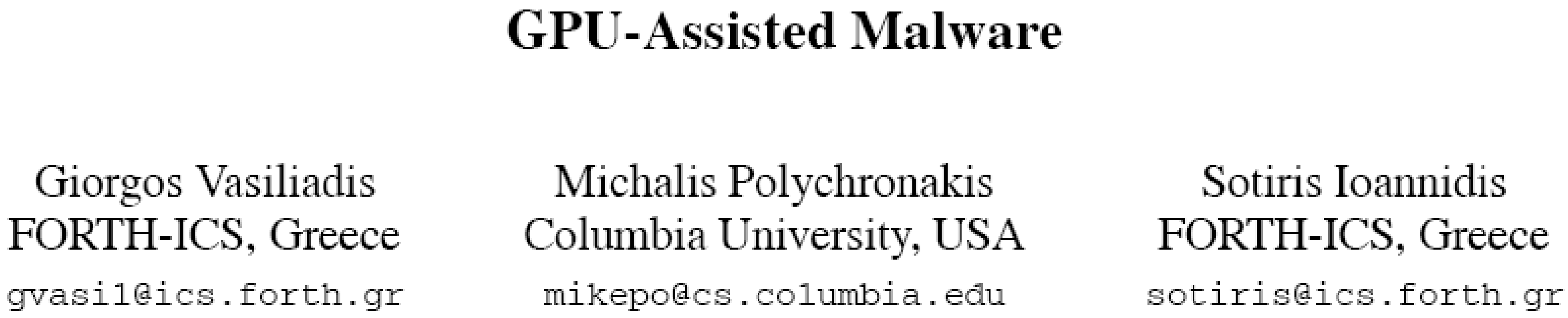
Obfuscation:

A mechanism employed by malware for concealing its presence on a system (e.g. utilizing the same name as a benign process, changing the last modified date of a malicious binary to that of a known binary)

Self-Defense:

A mechanism employed by malware that is intended to inhibit its removal after the successful infection of a system (e.g. the removal/shutdown of AV products, the disabling of specific services)

Malware writers are creative and resourceful !



GPU-Assisted Malware

Giorgos Vasiliadis
FORTH-ICS, Greece
gvasil@ics.forth.gr

Michalis Polychronakis
Columbia University, USA
mikepo@cs.columbia.edu

Sotiris Ioannidis
FORTH-ICS, Greece
sotiris@ics.forth.gr

Abstract

Malware writers constantly seek new methods to **obfuscate** their code so as to evade detection by virus scanners. Two **code-armoring** techniques that pose significant challenges to existing malicious-code detection and analysis systems are **unpacking** and **run-time polymorphism**. In this paper, we demonstrate how malware can increase its **robustness** against detection by taking advantage of the ubiquitous Graphics Processing Unit. We have designed and implemented unpacking and run-time polymorphism for a GPU, and tested them using existing graphics hardware. We also discuss how upcoming GPU features can be utilized to build even more robust, **evasive**, and functional malware.

Is it *really* that bad?

Fact 6

Application Hardening is the only means to lower the hacking ROI *and* increase the efficiency of online transactions.

Unfortunately, application hardening is still a largely misused concept, not easily quantifiable and measurable.

What’s a Hardened Application, anyway?

Any application capable of
withstanding attacks from sophisticated malware,
usable across several operating platforms,
easy to use and intuitive as a standard application,
loaded with all the best-of-breed protection techniques,
updated automatically and securely before each usage,
forcing hackers to work hard for each new attack.

Too vague! To be a little more specific:

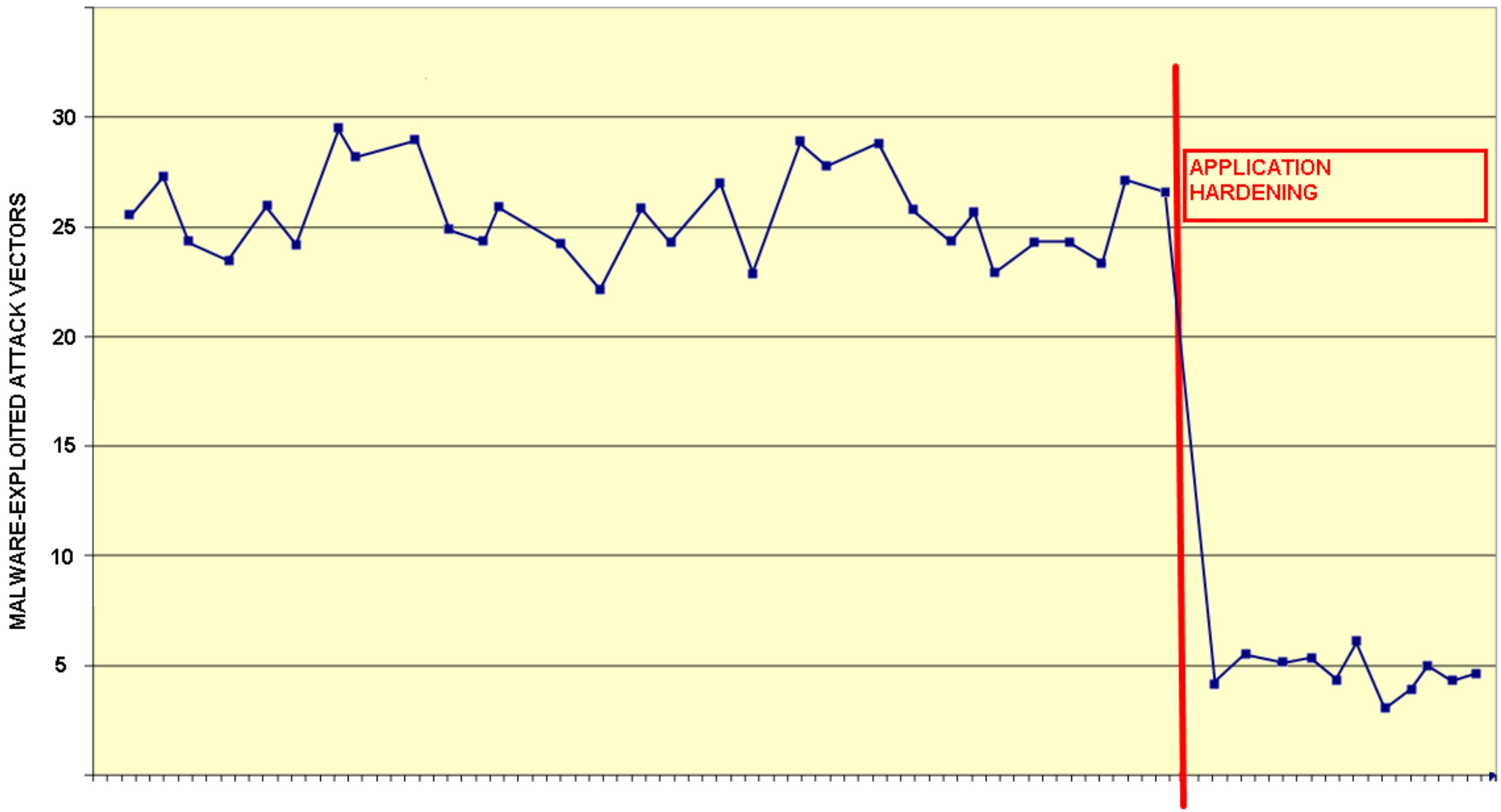
➢Executable Code

- employ a minimal build, reduce extensibility
- strip the code of all unnecessary components
- develop new secure modules, whenever possible
- permanently store exec codes, resource and config files in compressed, obfuscated and encrypted format
- unpack and load encrypted data at run time directly into memory only after passing strict integrity checks
- virtualize and morph critical executed code sections

➢Hardware Binding

- bind the executable to a tamper-proof device
- hide middleware and application configuration files
- use crypto chip to support server mutual authentication
- use crypto chip to enforce application authentication

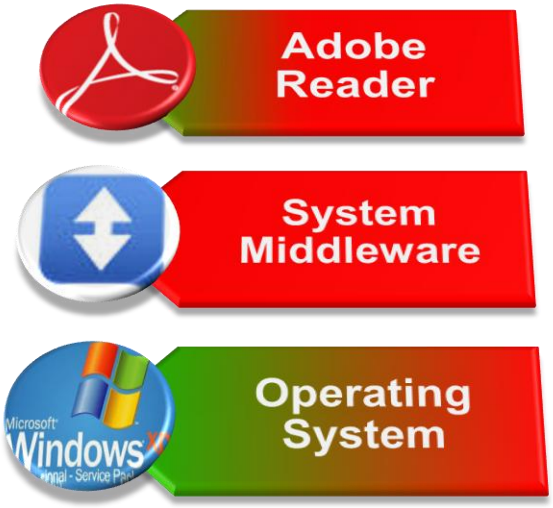
When Application Hardening Helps



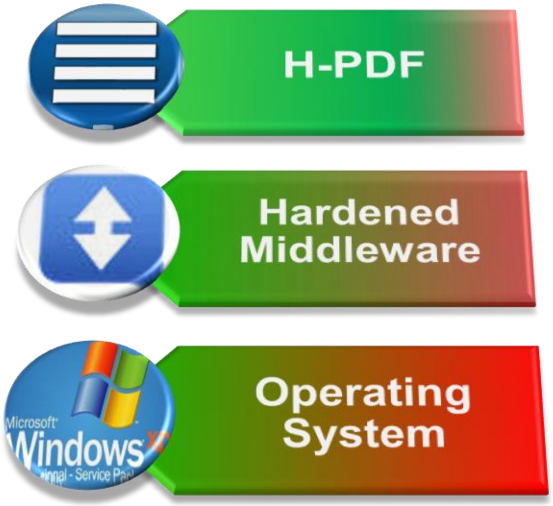
Measuring Security: Attack Vector Analysis

- Focus on attack vectors rather than on vulnerabilities, whereby the latter activate a **constant** set of attacks vectors which support malware.
- An attack vector is defined as *an elemental constituent of malware, necessary to enable at least one essential component of a malicious attack procedure*.
- In this usage, new vulnerabilities can replace old ineffective ones to enable the **same** attack vectors.
- AVA supports the principle that security must be **holistic** and enforced across all attack categories
- AVA helps the evaluation process of customers by providing a list of **indicators** directly correlated to the protection strength of any given hardening method

H-PDF™ : Example of Application Hardening



Adobe Reader®



H-PDF™

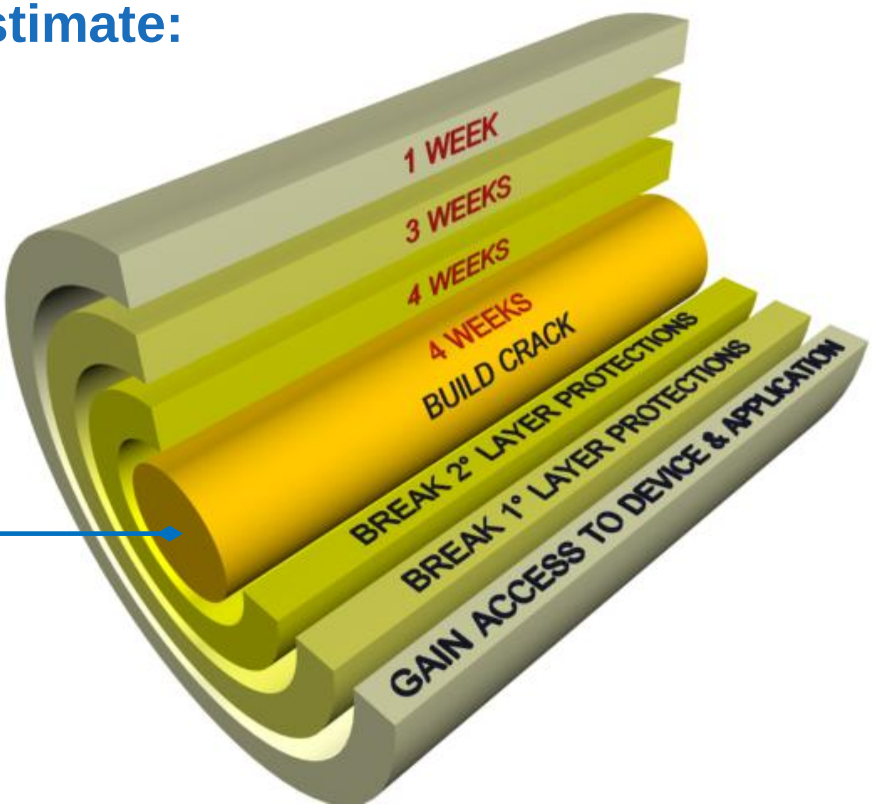
TRY TO KEEP’EM WORKING

PROACTIVE SECURITY updates can prevent the first-time effort hacking patches from being easily exploited for large scale attacks. The aim is to keep hacking success tied to ROI criteria.

This can be accomplished by using code **OBFUSCATION** techniques which are both **POLYMORPHIC** and **VIRTUALIZED**, so that periodic updates of the executable code will requrie hackers new efforts for building another attack code effective against the updated hardened application.

THE VALUE OF PROACTIVE SECURITY UPDATES

Example of Protection Layer Chafing Best Effort Estimate:


1 WEEK
3 WEEKS
4 WEEKS
4 WEEKS
BUILD CRACK
BREAK 2° LAYER PROTECTIONS
BREAK 1° LAYER PROTECTIONS
GAIN ACCESS TO DEVICE & APPLICATION

Even just one secure update per month will considerably complicate hacking efforts!

ANTI-VIRUSES COME TO THE RESCUE !

RELAX .... THEY ARE OURS
CANARY PETE

What went wrong and why?

SIGNATURE-BASED DETECTION

Obfuscation and run-time meta/poly-morphism

HEURISTIC-BASED DETECTION

Self-Defense (anti-reverse engineering, anti-debugging, etc.)

PROACTIVE DEFENSE

Process protection (anti-injection, anti-vitualization, etc.)

SNAPSHOT FROM THE BATTLE FIELD…..

...WE'RE TAKING FIRE!
IT'S OK. SARGE... IT'S FRIENDLY FIRE..
GRESZ.

IS THIS ALL? CALL IT FFAPI OR WHATEVER….

- The burden of maintaining a sustainable hardened application environment cannot be placed only on the ISVs.
- We need to develop a structured and coordinated procedure allowing AV products and hardened applications to mutually interrogate each other in order to discriminate legitimate tasks from potentially hostile processes:

1. ISV submits a valid digital certificates to FFAPI.
2. FFAPI records the certificate’s fingerprint, validity dates, etc, in a centralized database accessible by all AV vendors.
3. The AV software probes the ISV software for threats by checking the digital signature of the executable code, the DLLs, the resource files, etc. Any unsigned component is blacklisted.
4. All other checks are bypassed if the previous step is successful.

CONCLUSIONS

- Fighting the malware epidemic requires a shift from defensive to pro-active security, forcing hackers to work for each new attack and restricting the number of viable attack vectors.
- General purpose applications cannot achieve acceptable levels of security. End-users cannot properly install, configure and maintain a secure computing environment.
- Architectural hardening is essential to make attacks very complex and to reduce the strength and variety of social engineering attacks down to physiological fraud levels. Hardening must be balanced and extensive.
- A coordinated effort among AV vendors and software companies is required to make application hardening sustainable and scalable.
- Unless such an effort is successful, it will not be possible to effectively secure online transactions when AV protections fail.

THANK YOU!

FOR FURTHER INFORMATION: **cronchi@eisst.com**

EISST Ltd
Fairfax House,
15 Fulwood Place
London WC1V 6AY, UK
T: +44 (0)20 79 695 688
F: +44 (0)20 77 483 273
E: info@eisst.com
W: www.eisst.com