CARO2010 HELSINKI

Mario Vuksan and Tomislav Pericin, ReversingLabs

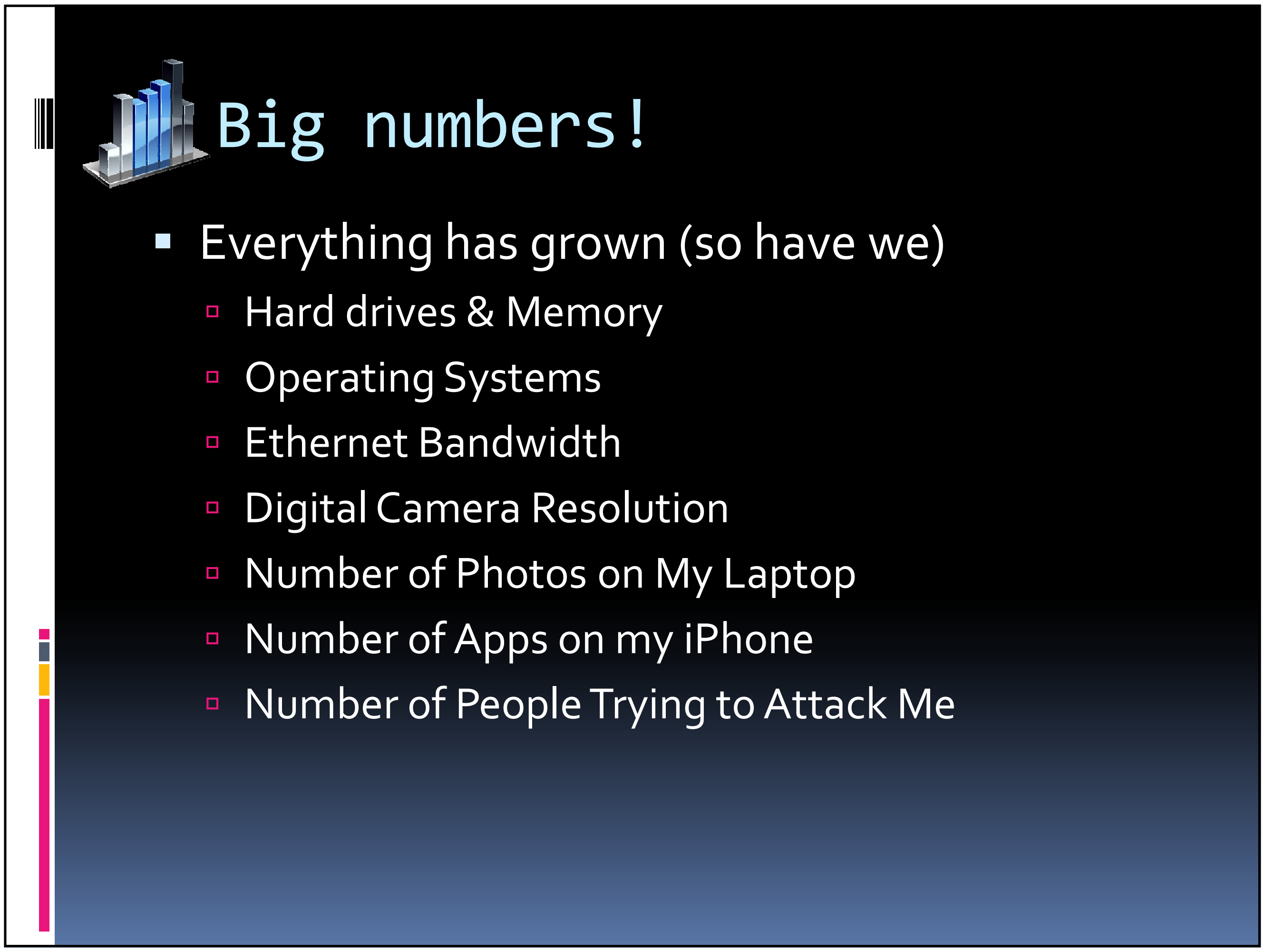
FILE ANALYSIS AND UNPACKING:
THE AGE OF 40M NEW SAMPLES PER YEAR

Agenda

- Big and scary numbers
- Introduction to the binary mess out there (the problem)
  - Packers and compressions everywhere we look
- Introduction to the arsenal of file analysis tools
  - Identification, validation and unpacking
- Introduction to the binary layering (the solution)
  - Better identification, validation and unpacking

REVERSING LABS | Reverse Engineering & Software Protection

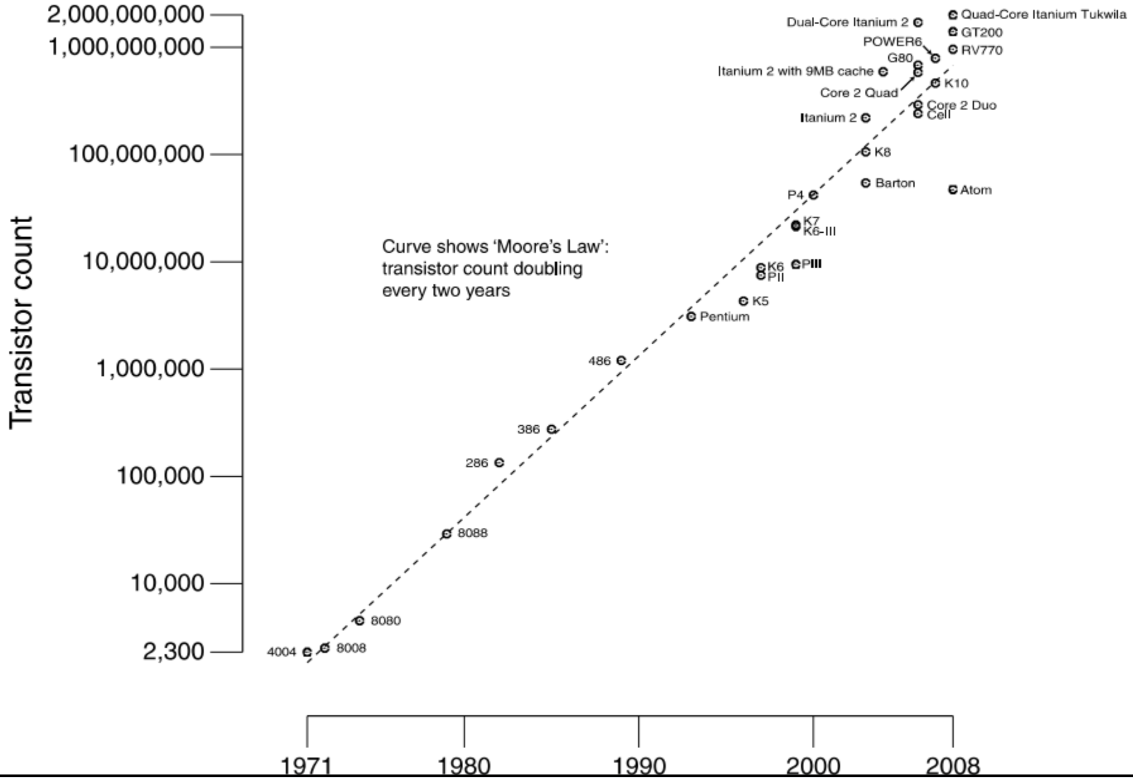
CARO2010 HELSINKI



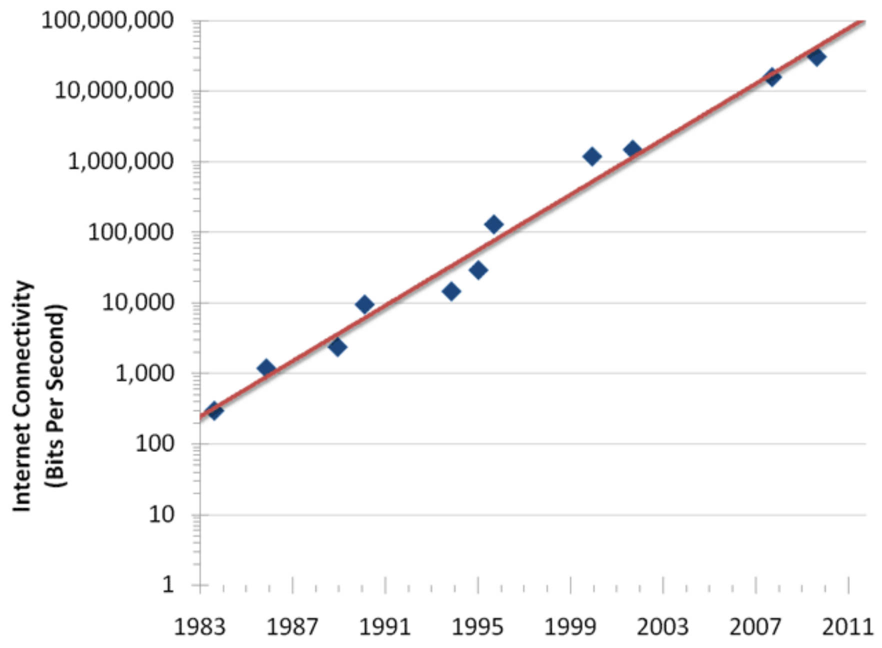
Big numbers!

- Everything has grown (so have we)
  - Hard drives & Memory
  - Operating Systems
  - Ethernet Bandwidth
  - Digital Camera Resolution
  - Number of Photos on My Laptop
  - Number of Apps on my iPhone
  - Number of People Trying to Attack Me

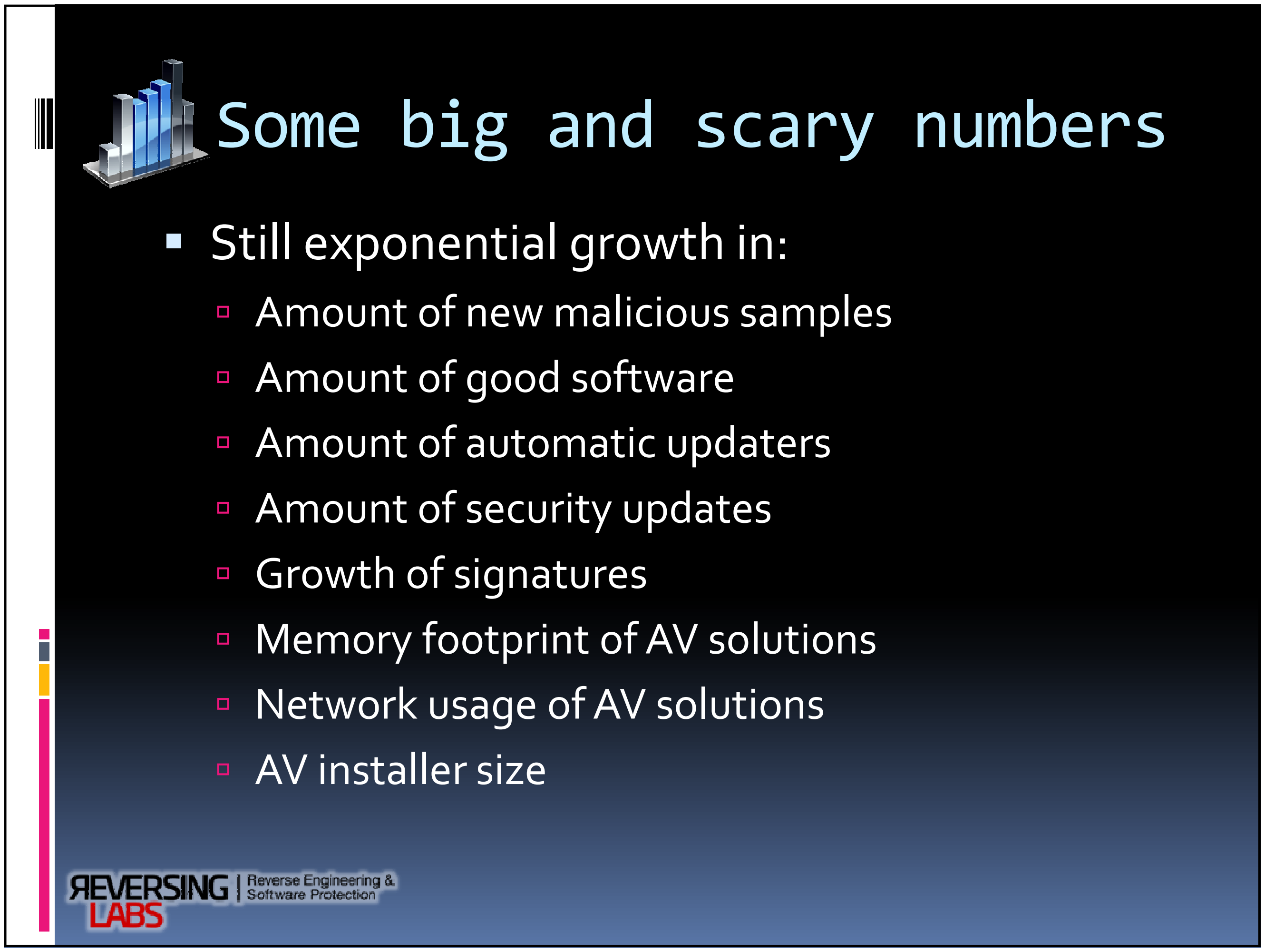
Moore’s Law – 60% Growth



Nielsen’s Law – 50% Growth



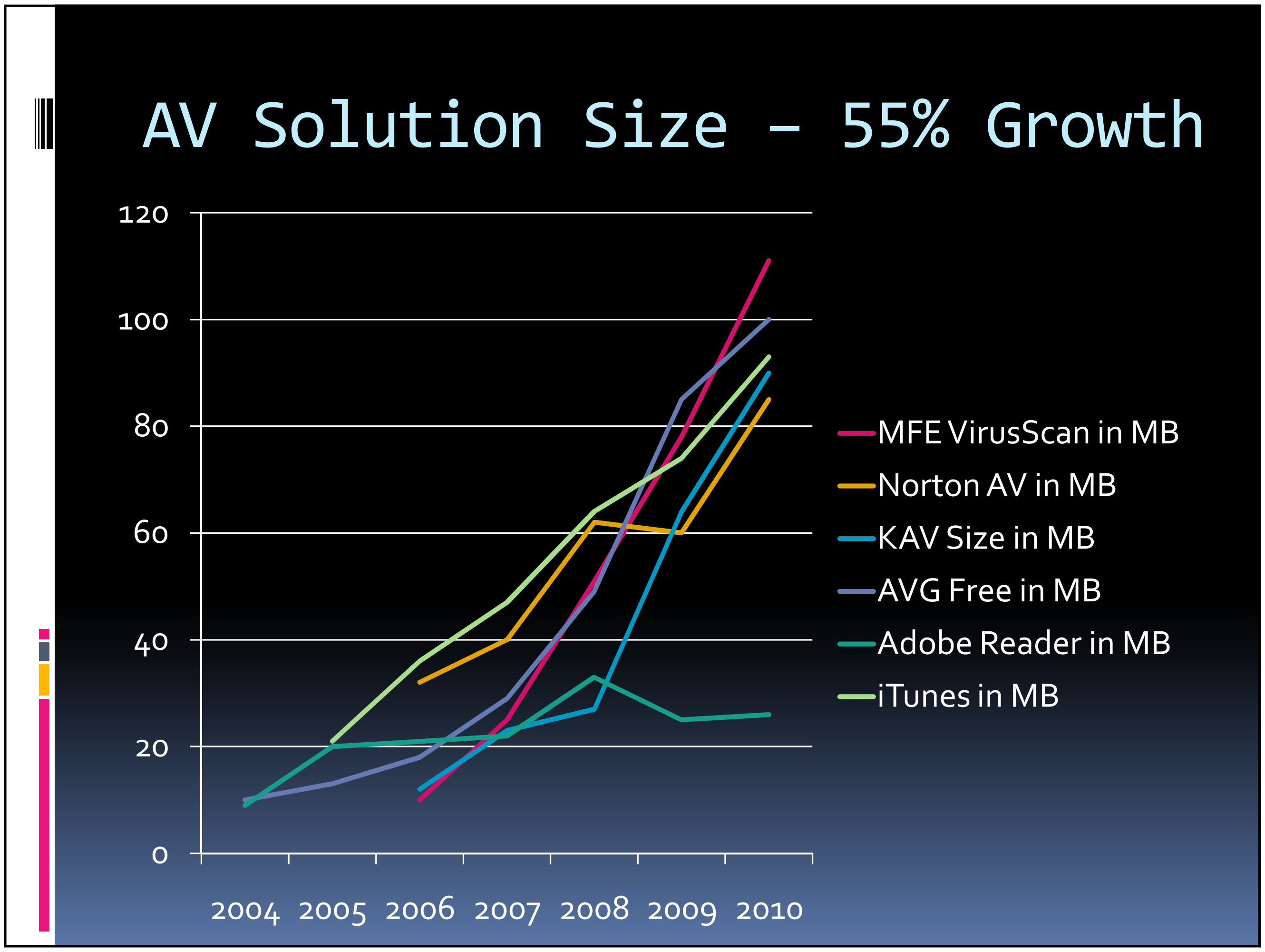
Source: Jakob Nielsen



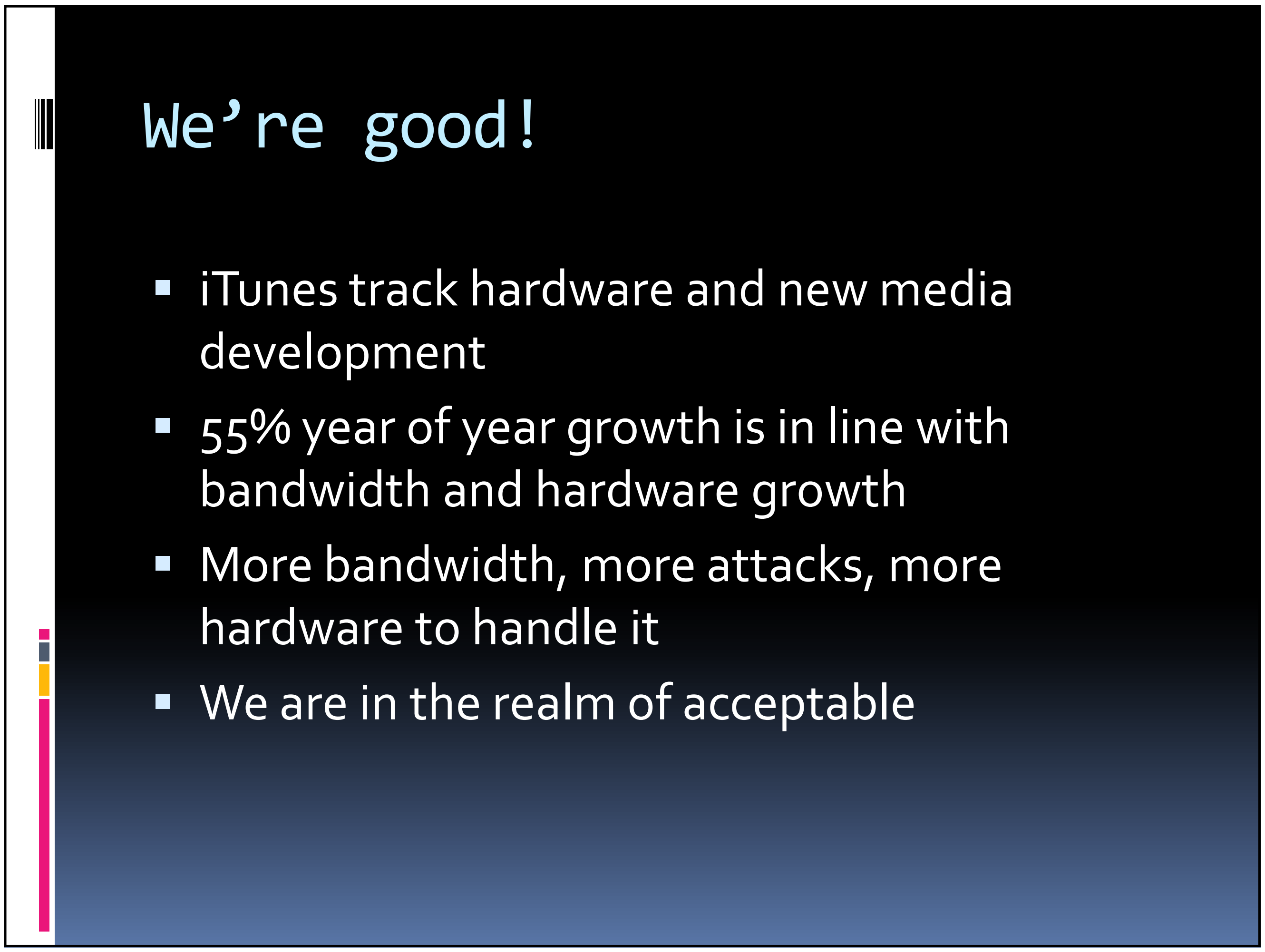
Some big and scary numbers

- Still exponential growth in:
  - Amount of new malicious samples
  - Amount of good software
  - Amount of automatic updaters
  - Amount of security updates
  - Growth of signatures
  - Memory footprint of AV solutions
  - Network usage of AV solutions
  - AV installer size

REVERSING LABS | Reverse Engineering & Software Protection

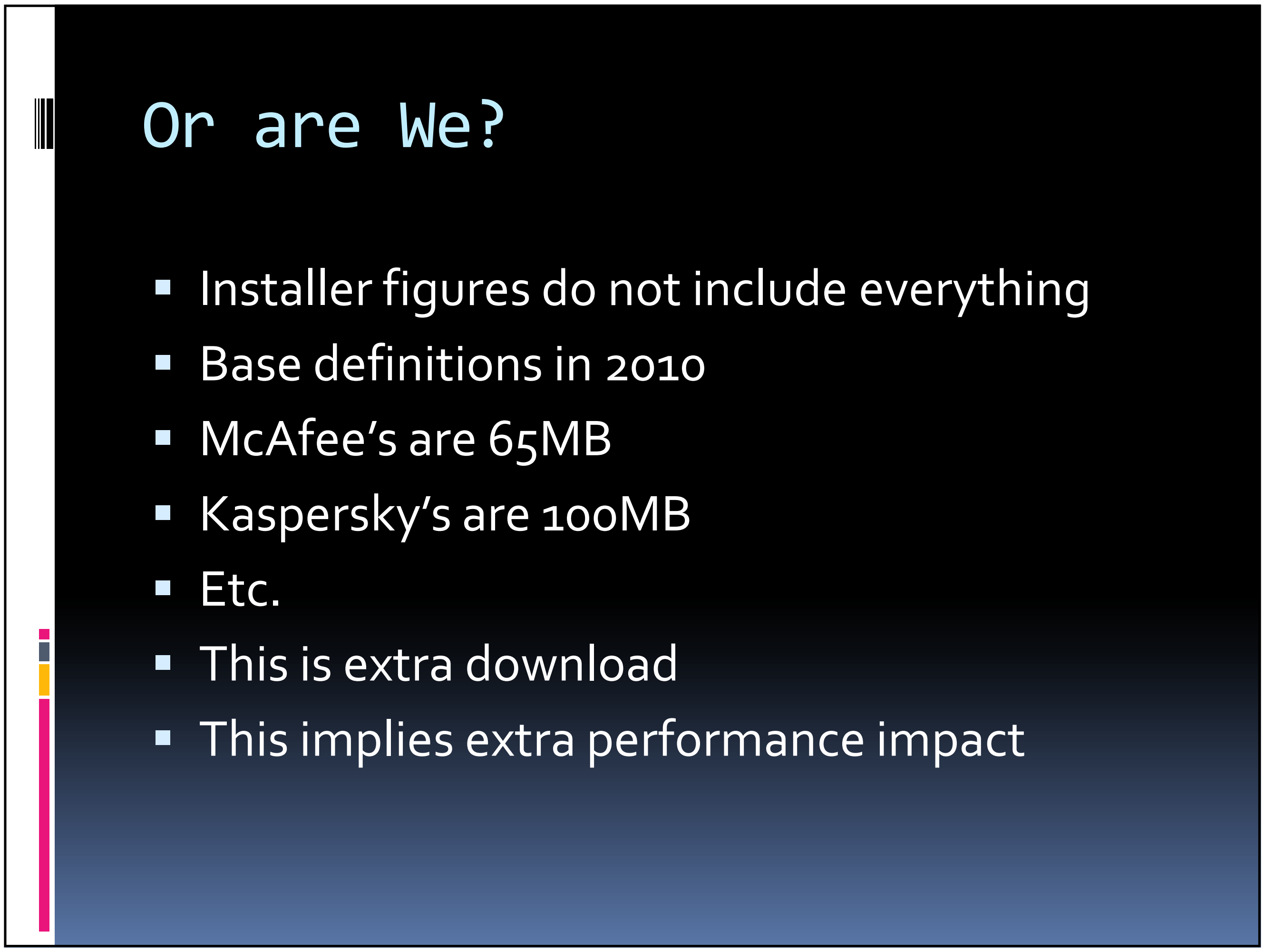


AV Solution Size – 55% Growth



We're good!

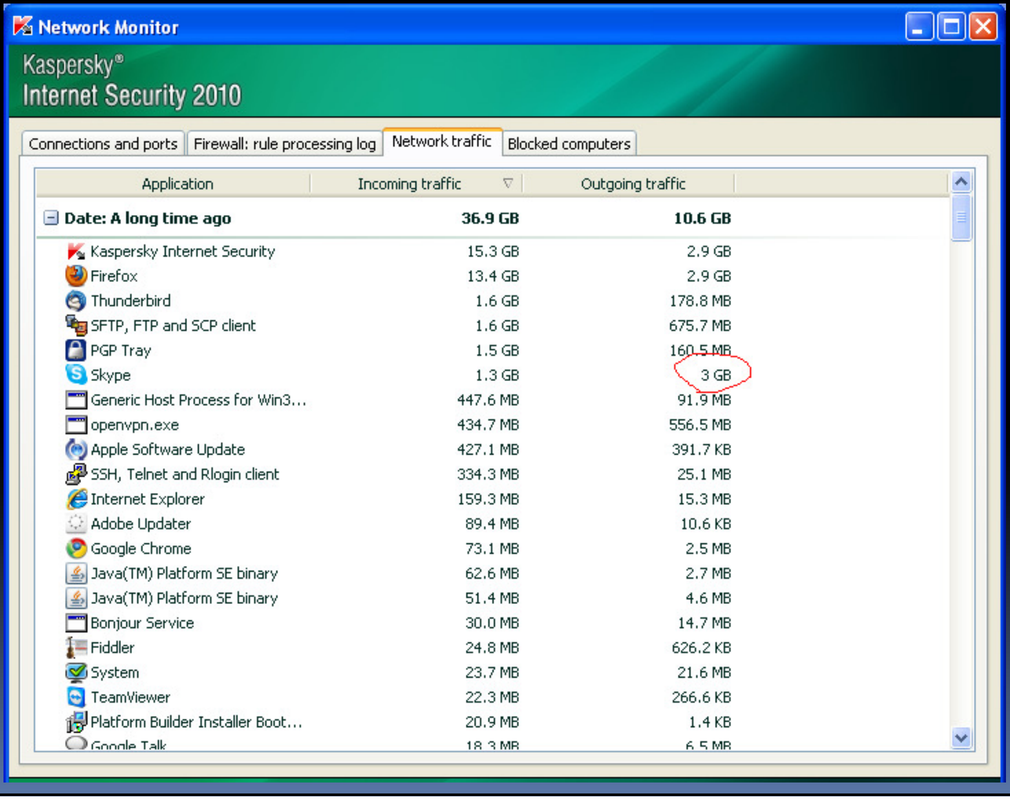
- iTunes track hardware and new media development
- 55% year of year growth is in line with bandwidth and hardware growth
- More bandwidth, more attacks, more hardware to handle it
- We are in the realm of acceptable



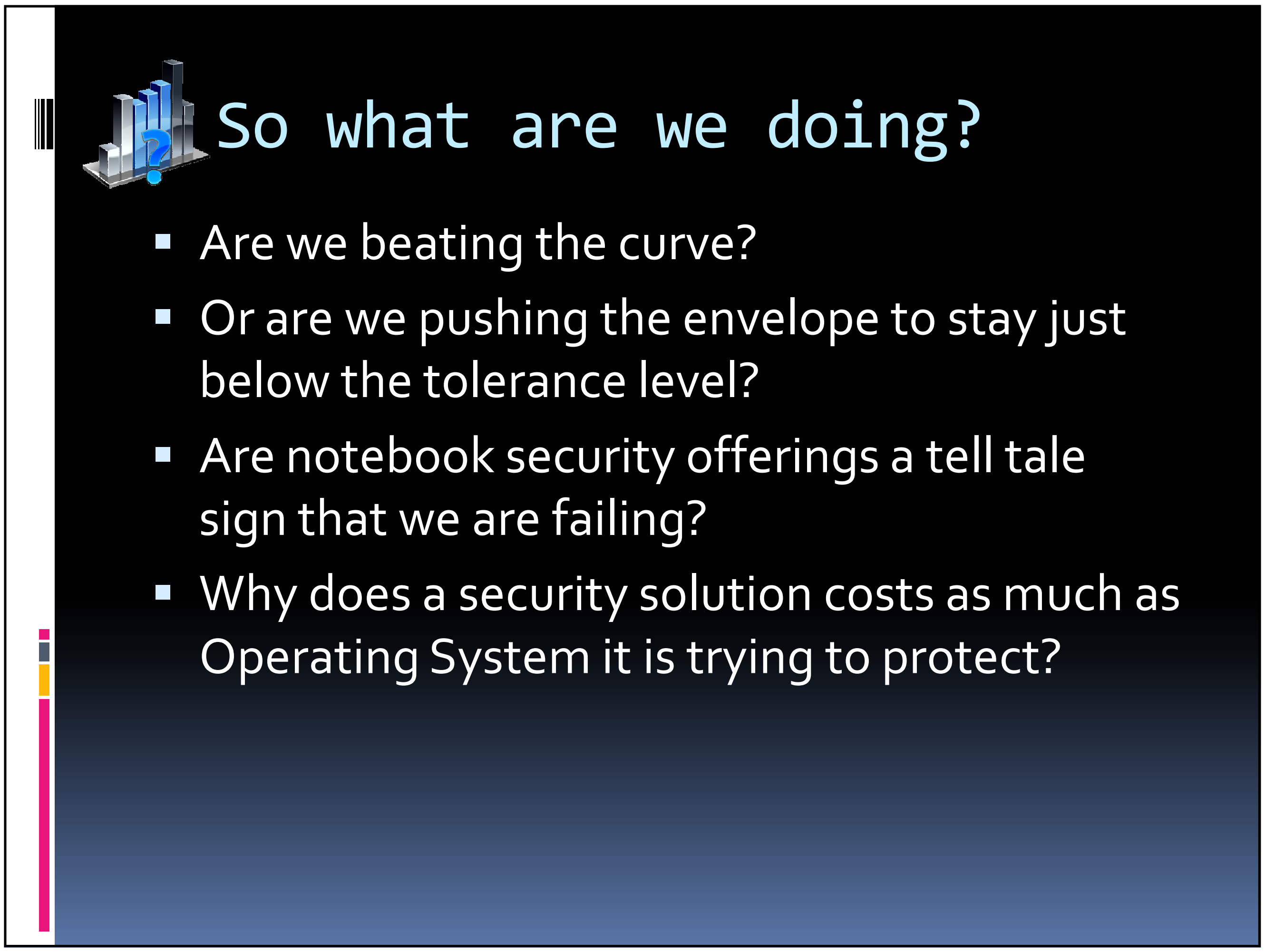
Or are We?

- Installer figures do not include everything
- Base definitions in 2010
- McAfee's are 65MB
- Kaspersky's are 100MB
- Etc.
- This is extra download
- This implies extra performance impact

And Extra Bandwidth Usage

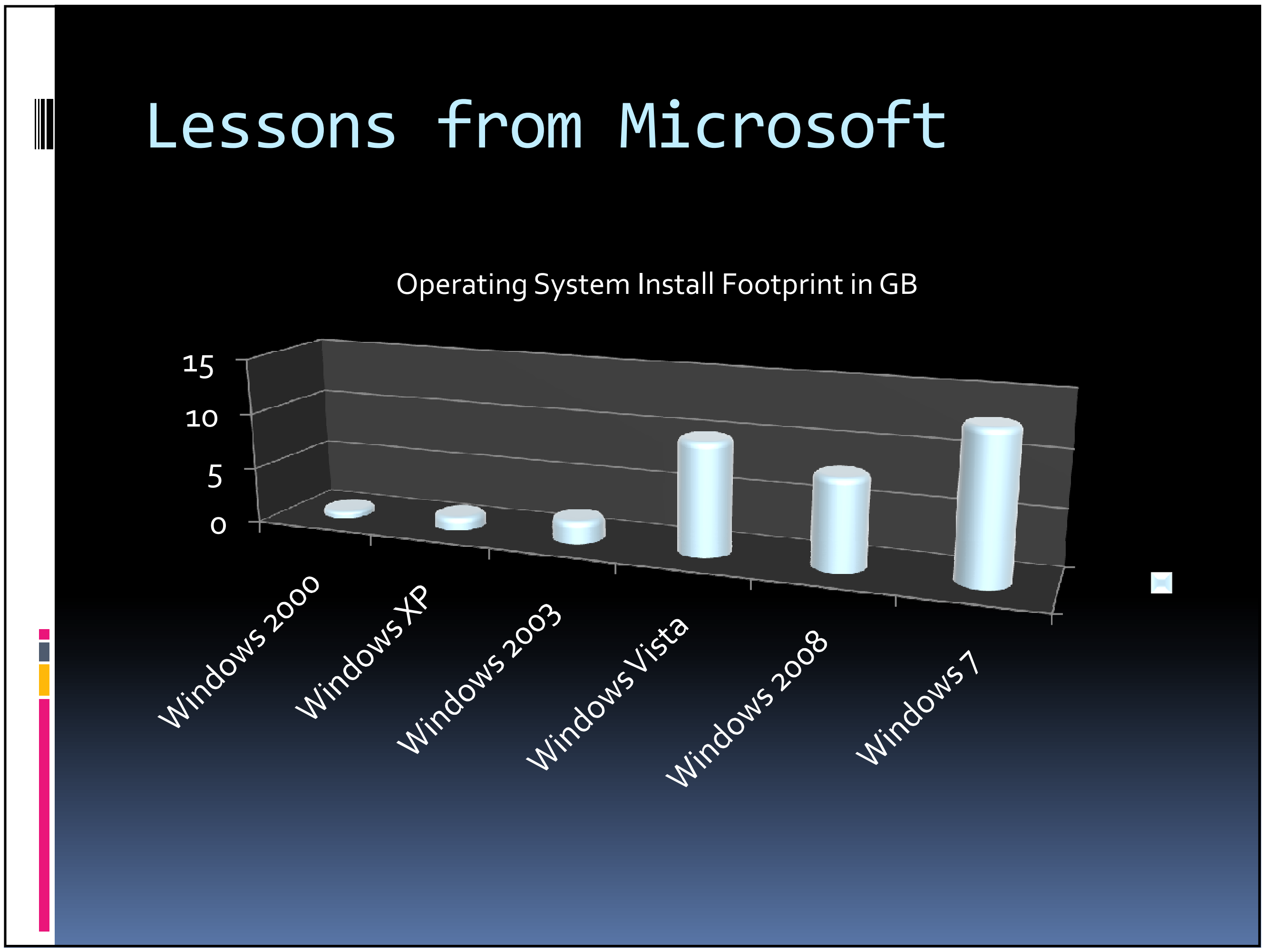


Application	Incoming traffic	Outgoing traffic
Date: A long time ago	**36.9 GB**	**10.6 GB**
Kaspersky Internet Security	15.3 GB	2.9 GB
Firefox	13.4 GB	2.9 GB
Thunderbird	1.6 GB	178.8 MB
SFTP, FTP and SCP client	1.6 GB	675.7 MB
PGP Tray	1.5 GB	160.5 MB
Skype	1.3 GB	3 GB
Generic Host Process for Win3...	447.6 MB	91.9 MB
openvpn.exe	434.7 MB	556.5 MB
Apple Software Update	427.1 MB	391.7 KB
SSH, Telnet and Rlogin client	334.3 MB	25.1 MB
Internet Explorer	159.3 MB	15.3 MB
Adobe Updater	89.4 MB	10.6 KB
Google Chrome	73.1 MB	2.5 MB
Java(TM) Platform SE binary	62.6 MB	2.7 MB
Java(TM) Platform SE binary	51.4 MB	4.6 MB
Bonjour Service	30.0 MB	14.7 MB
Fiddler	24.8 MB	626.2 KB
System	23.7 MB	21.6 MB
TeamViewer	22.3 MB	266.6 KB
Platform Builder Installer Boot...	20.9 MB	1.4 KB
Google Talk	18.3 MB	6.5 MB

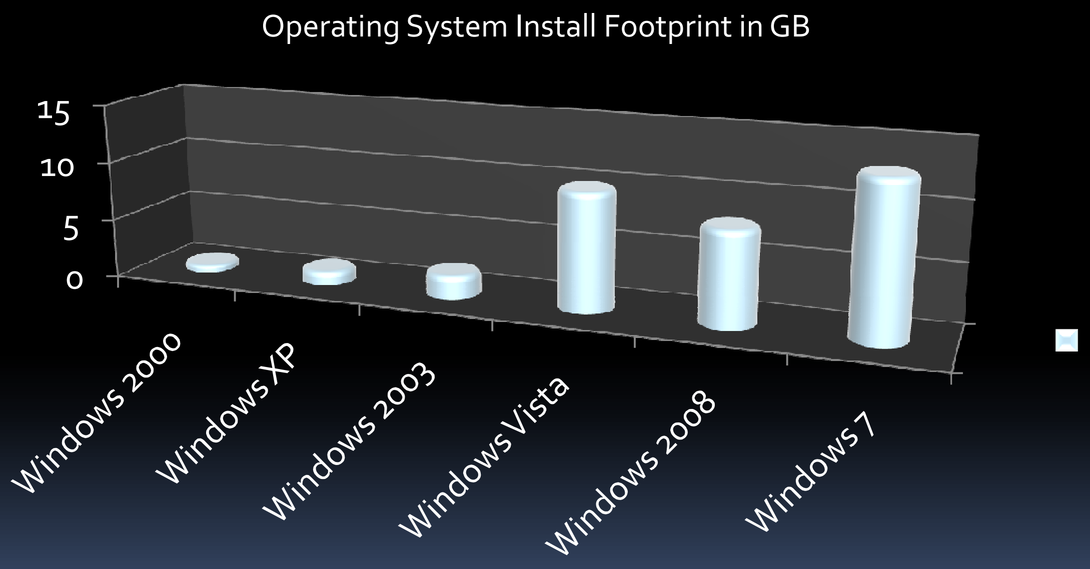


So what are we doing?

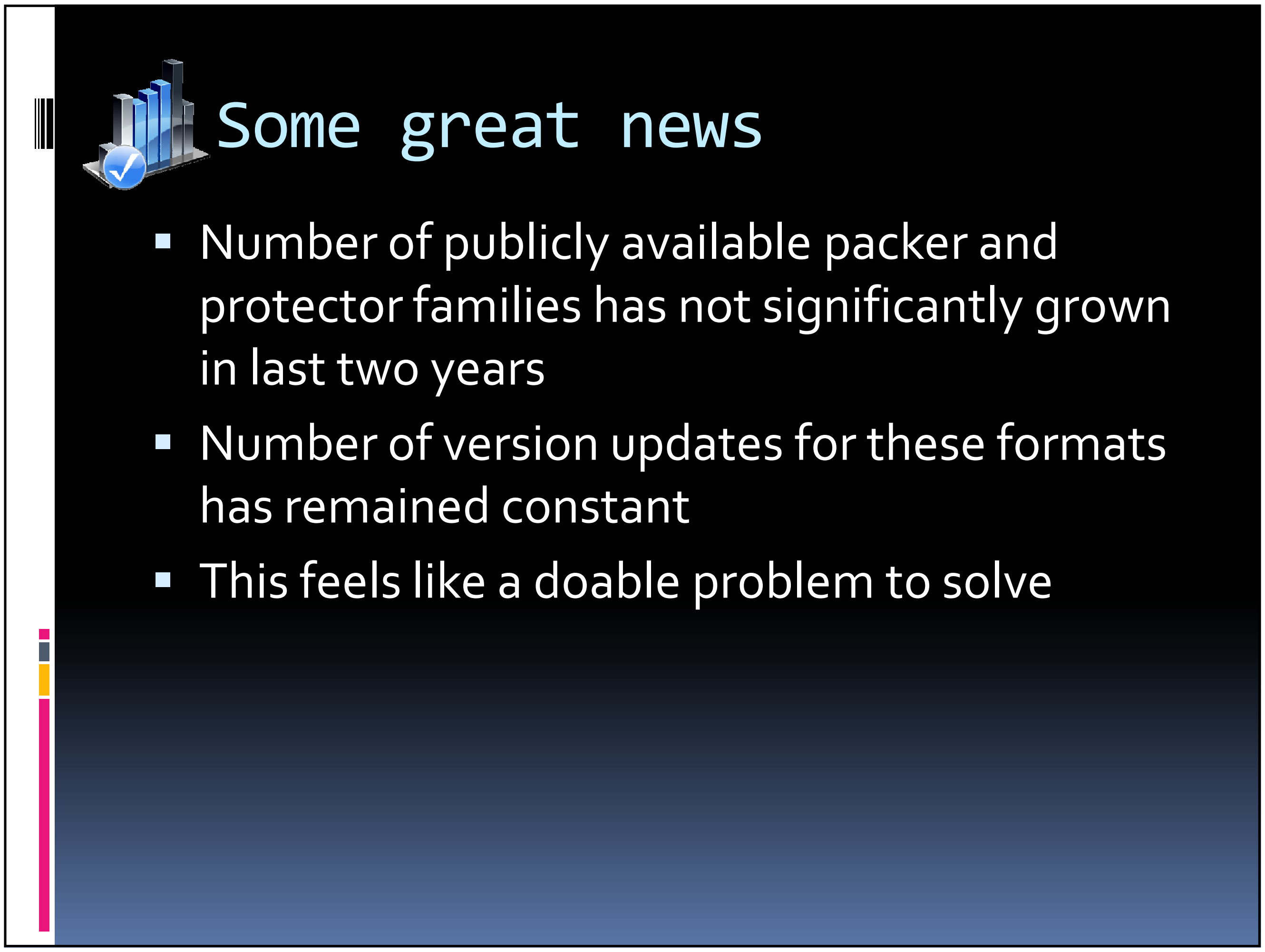
- Are we beating the curve?
- Or are we pushing the envelope to stay just below the tolerance level?
- Are notebook security offerings a tell tale sign that we are failing?
- Why does a security solution costs as much as Operating System it is trying to protect?



Lessons from Microsoft

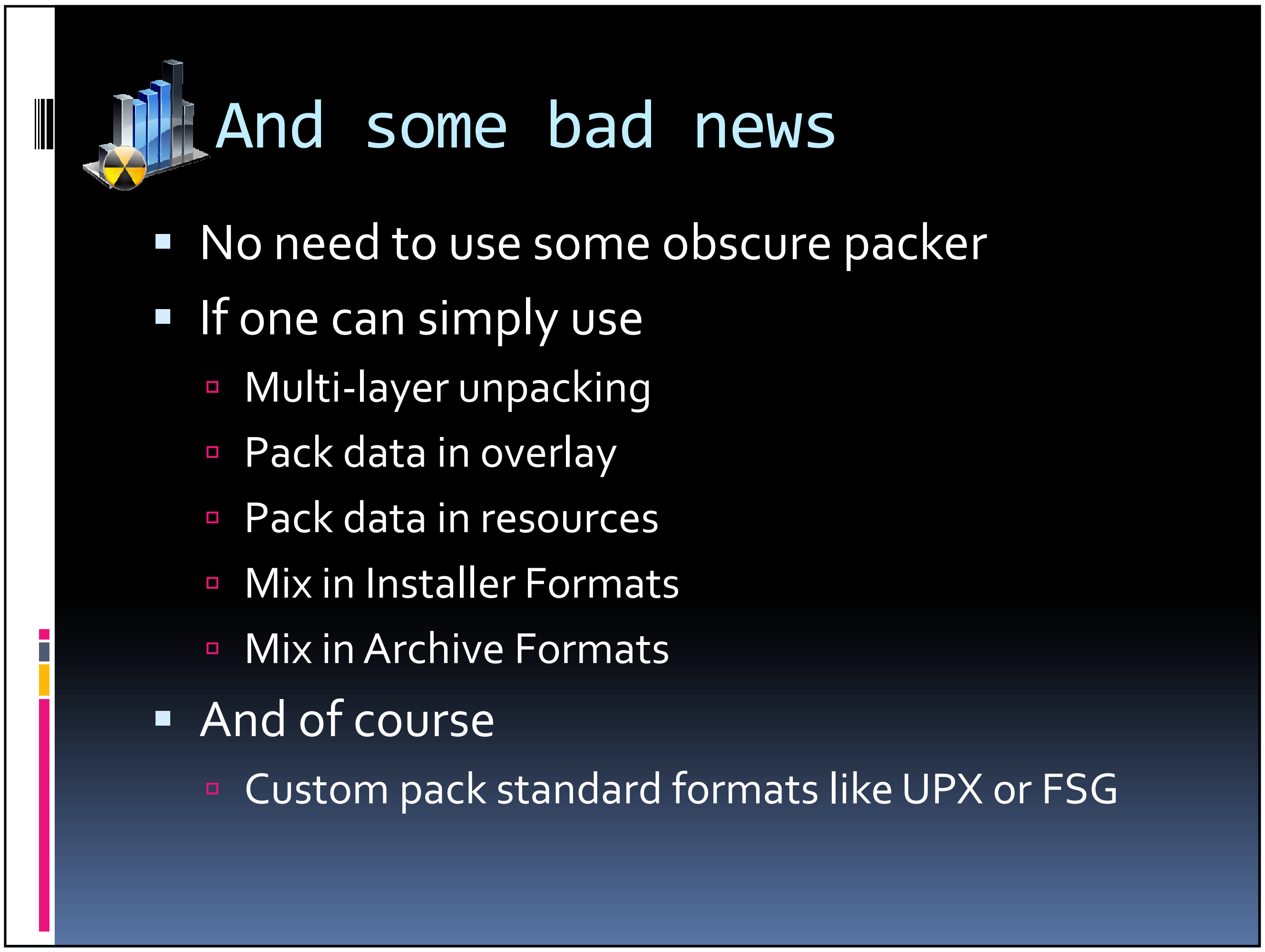


File Analysis & Unpacking



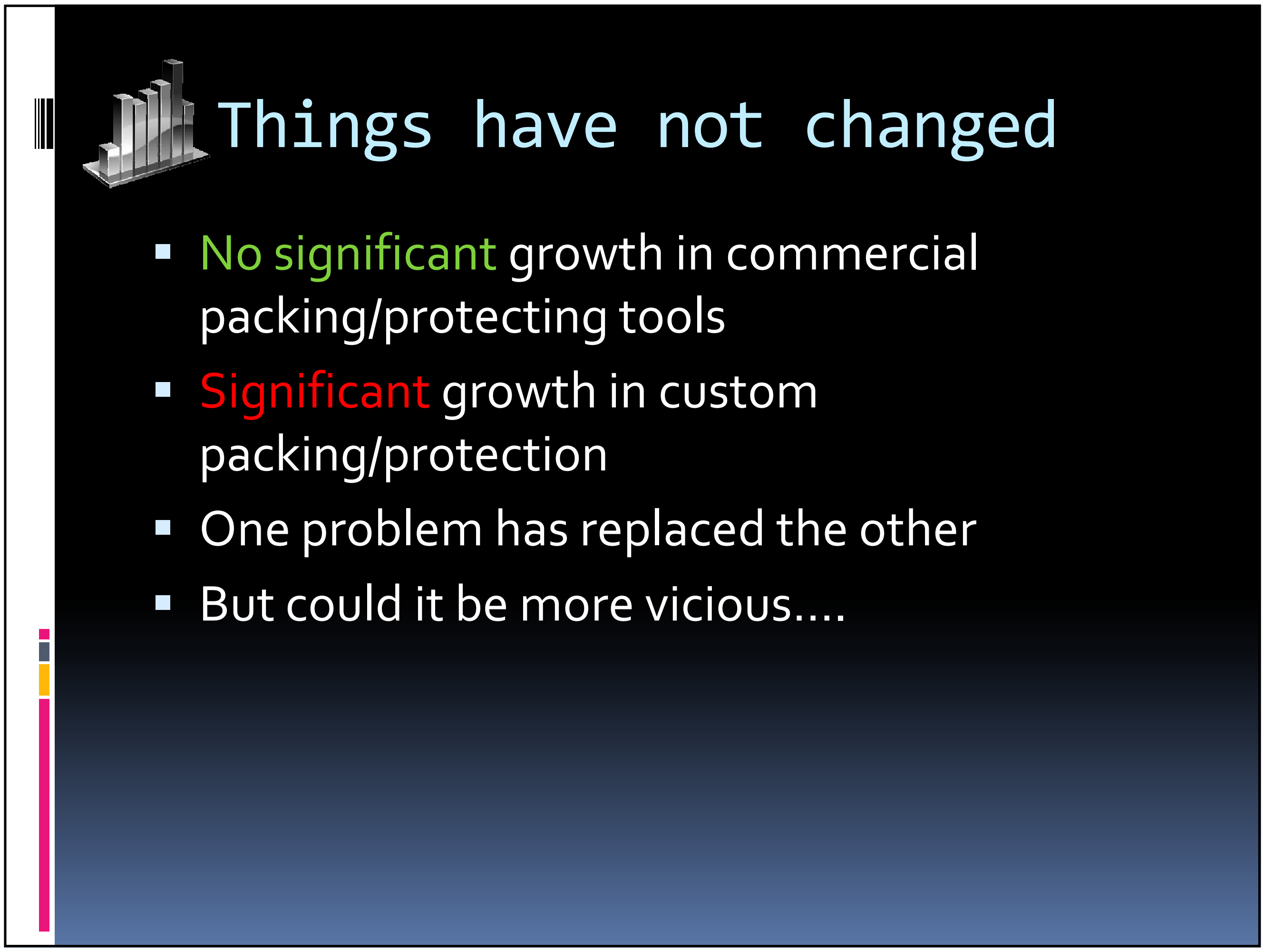
Some great news

- Number of publicly available packer and protector families has not significantly grown in last two years
- Number of version updates for these formats has remained constant
- This feels like a doable problem to solve



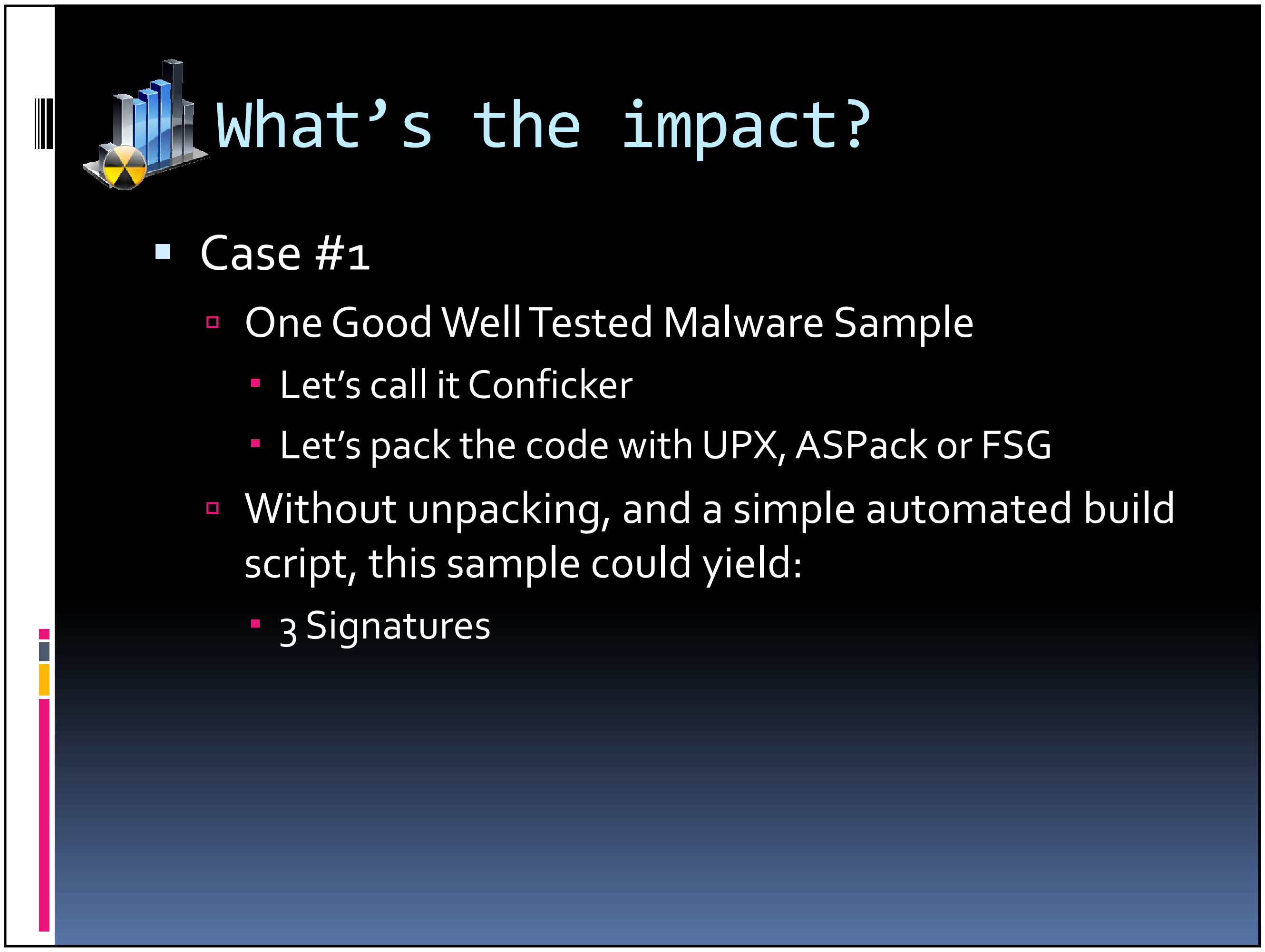
And some bad news

- No need to use some obscure packer
- If one can simply use
  - Multi-layer unpacking
  - Pack data in overlay
  - Pack data in resources
  - Mix in Installer Formats
  - Mix in Archive Formats
- And of course
  - Custom pack standard formats like UPX or FSG



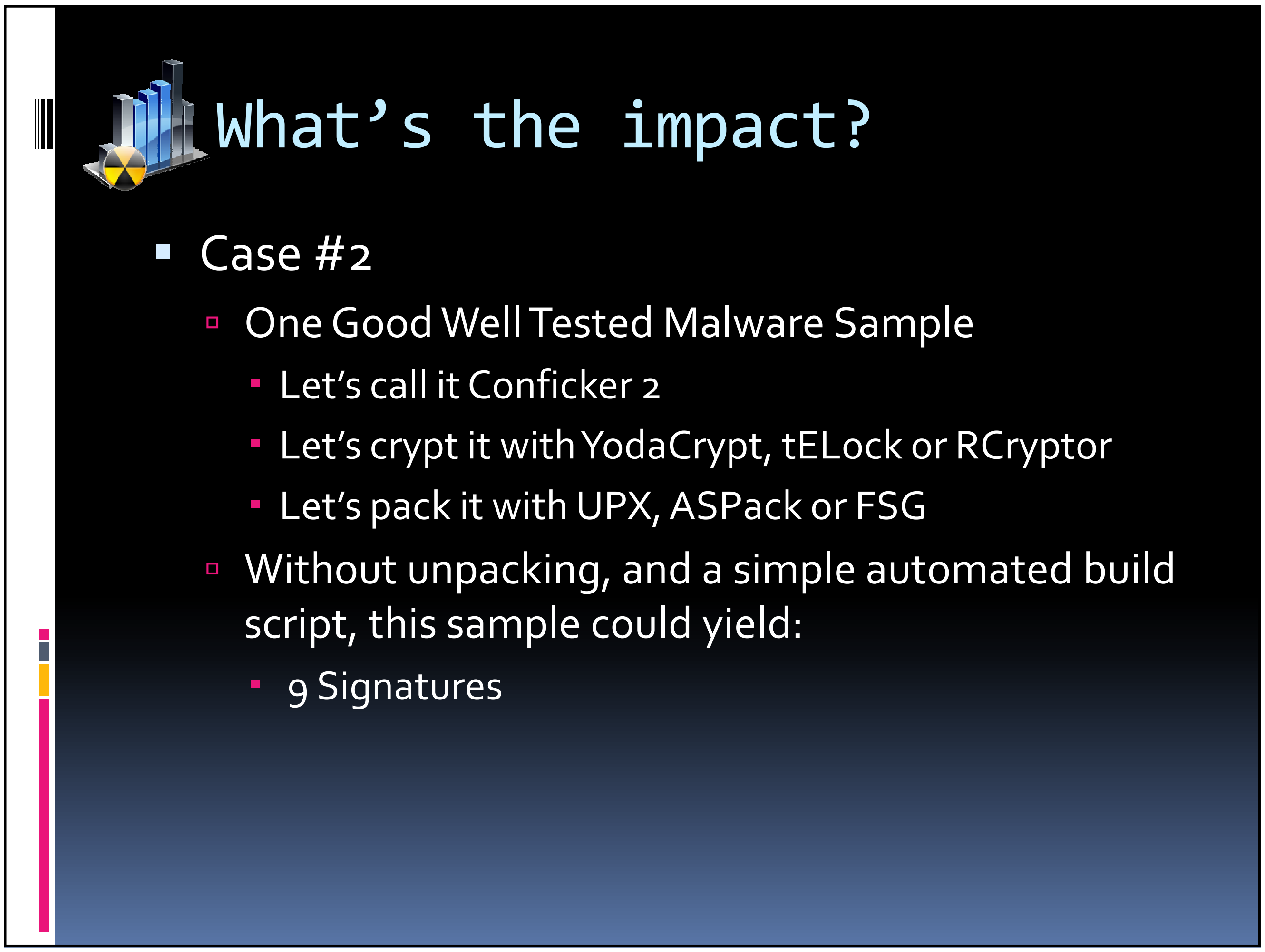
Things have not changed

- No significant growth in commercial packing/protecting tools
- Significant growth in custom packing/protection
- One problem has replaced the other
- But could it be more vicious….



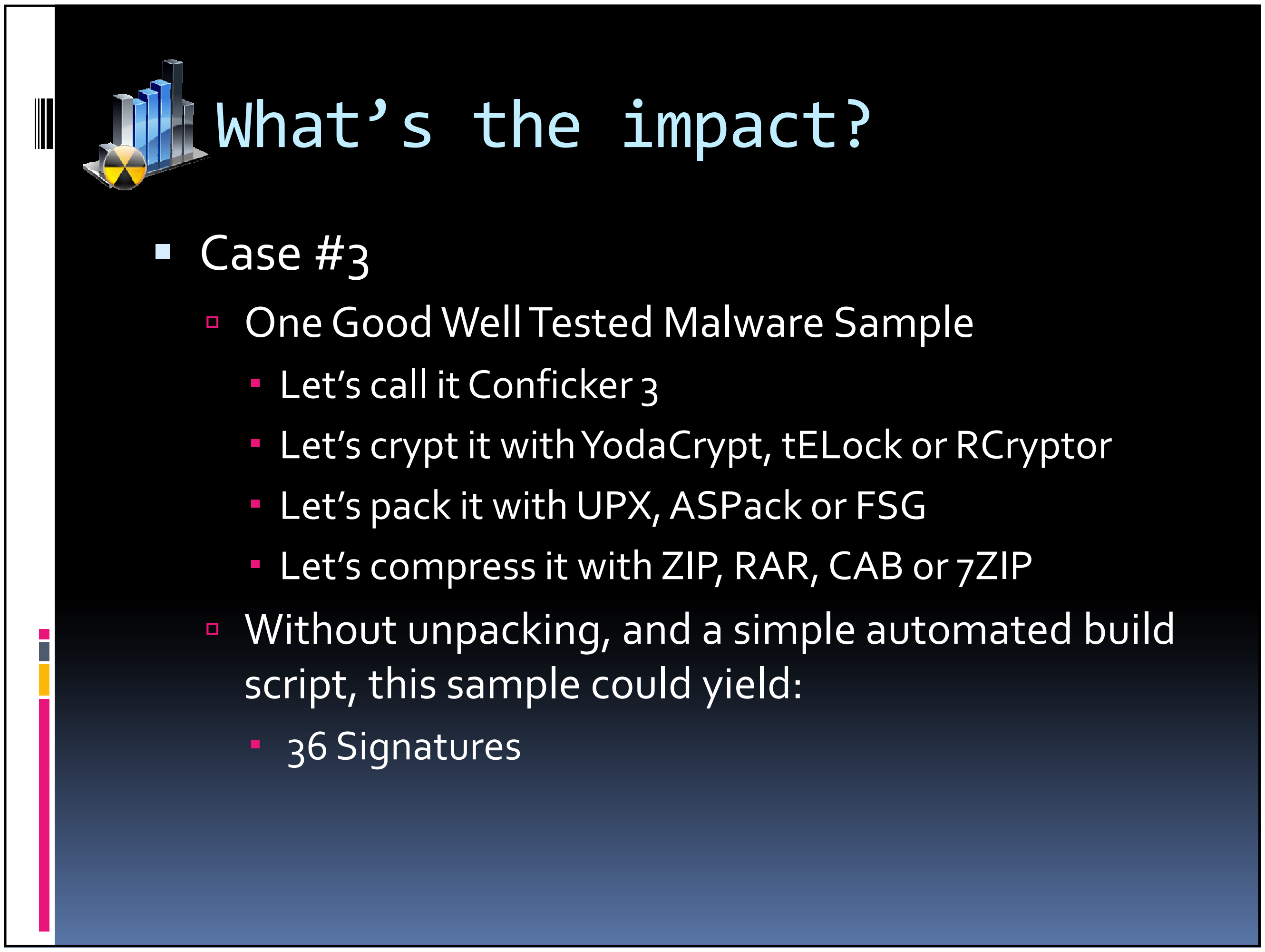
What’s the impact?

- Case #1
  - One Good Well Tested Malware Sample
    - Let’s call it Conficker
    - Let’s pack the code with UPX, ASPack or FSG
  - Without unpacking, and a simple automated build script, this sample could yield:
    - 3 Signatures



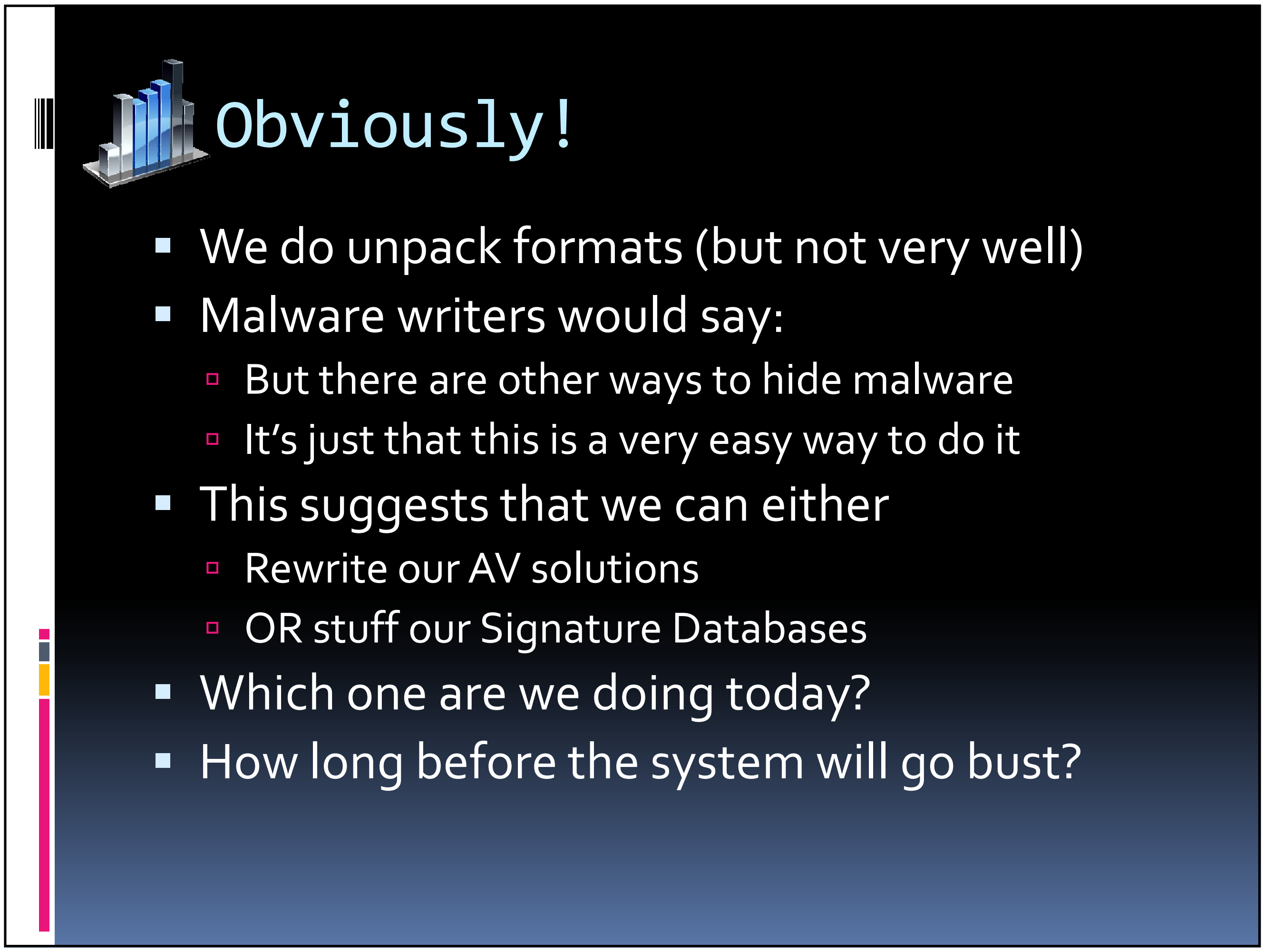
What's the impact?

- Case #2
  - One Good Well Tested Malware Sample
    - Let's call it Conficker 2
    - Let's crypt it with YodaCrypt, tELock or RCryptor
    - Let's pack it with UPX, ASPack or FSG
  - Without unpacking, and a simple automated build script, this sample could yield:
    - 9 Signatures



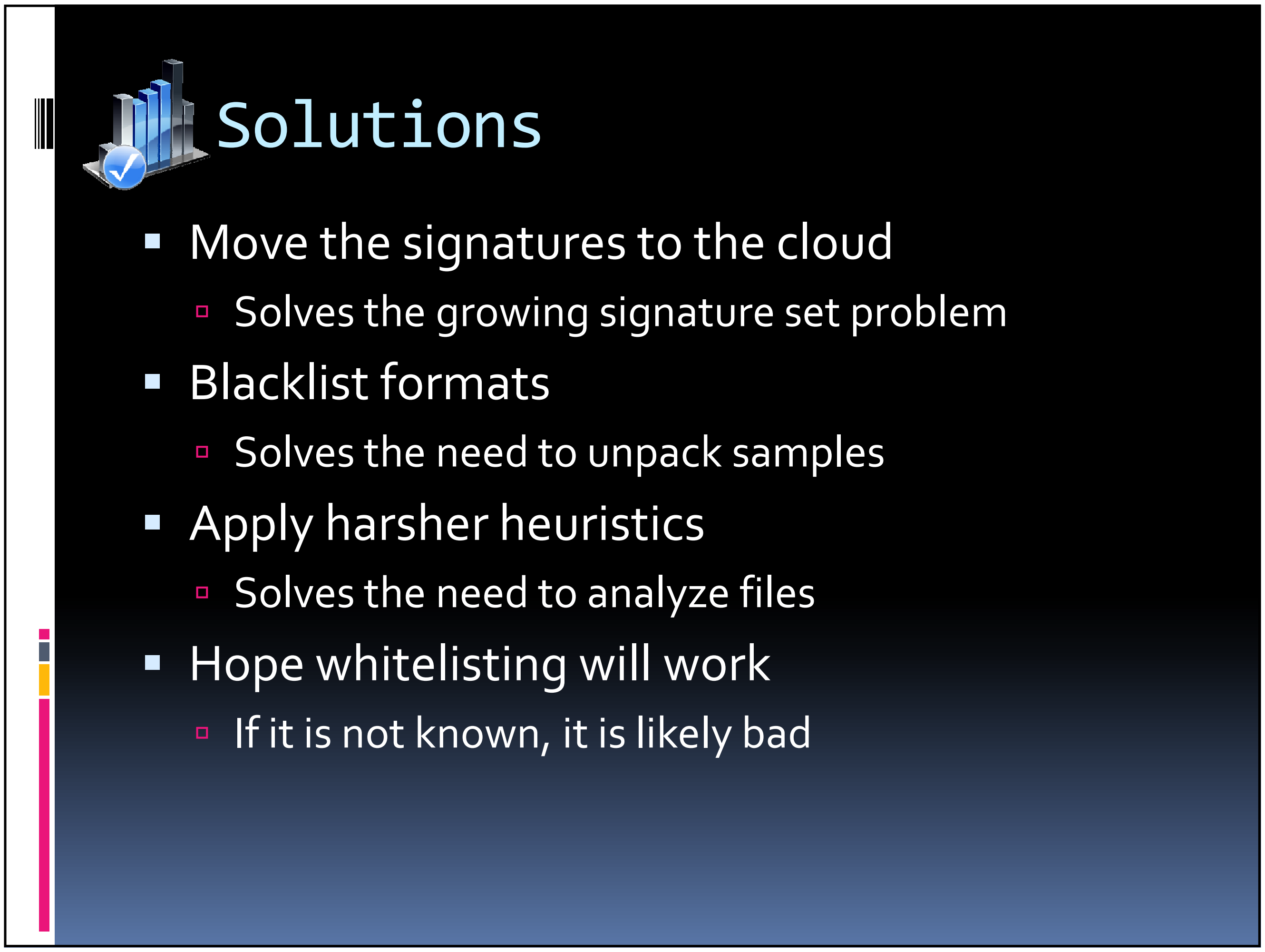
What's the impact?

- Case #3
  - One Good Well Tested Malware Sample
    - Let's call it Conficker 3
    - Let's crypt it with YodaCrypt, tELock or RCryptor
    - Let's pack it with UPX, ASPack or FSG
    - Let's compress it with ZIP, RAR, CAB or 7ZIP
  - Without unpacking, and a simple automated build script, this sample could yield:
    - 36 Signatures



Obviously!

- We do unpack formats (but not very well)
- Malware writers would say:
  - But there are other ways to hide malware
  - It's just that this is a very easy way to do it
- This suggests that we can either
  - Rewrite our AV solutions
  - OR stuff our Signature Databases
- Which one are we doing today?
- How long before the system will go bust?

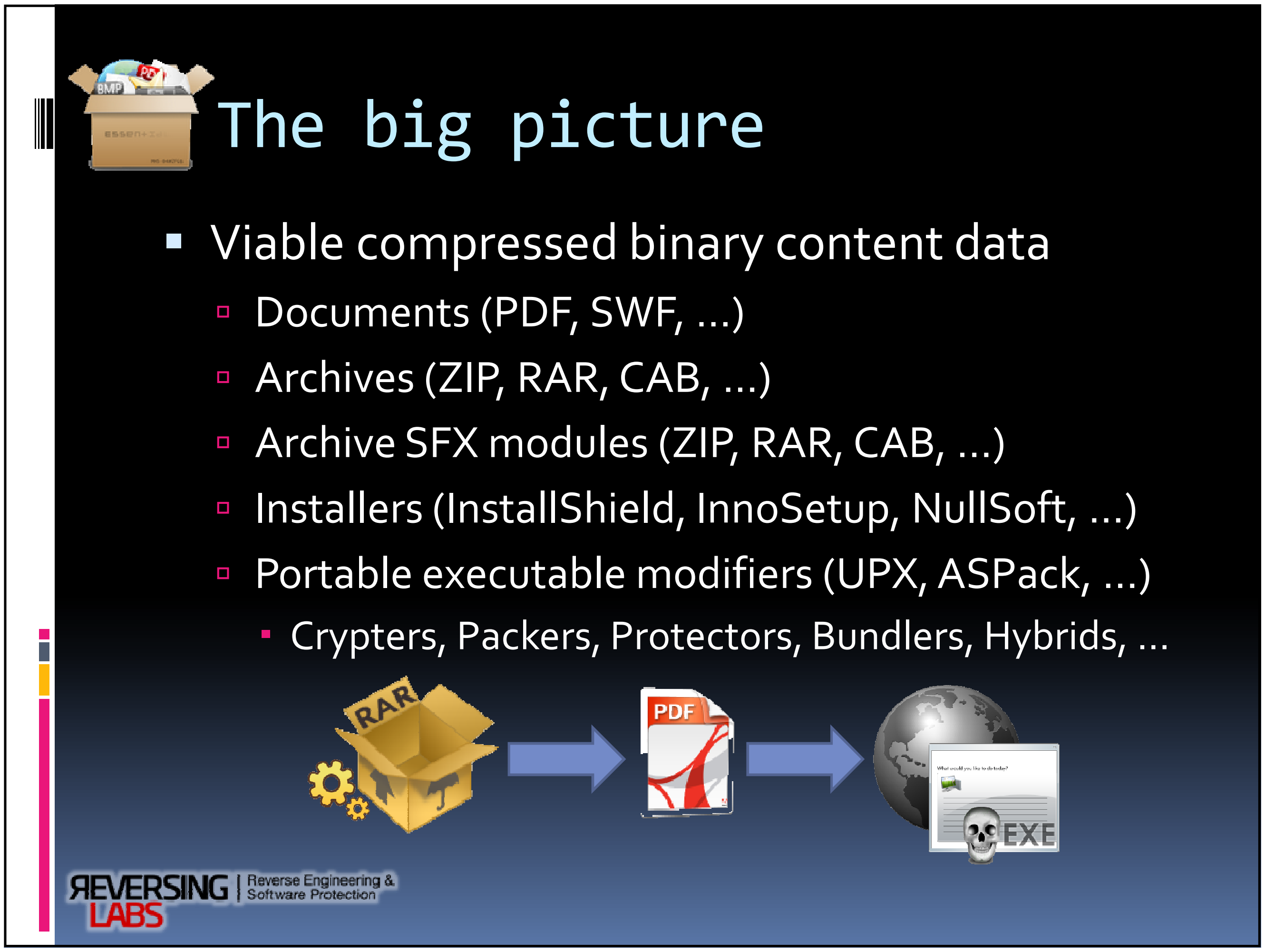


Solutions

- Move the signatures to the cloud
  - Solves the growing signature set problem
- Blacklist formats
  - Solves the need to unpack samples
- Apply harsher heuristics
  - Solves the need to analyze files
- Hope whitelisting will work
  - If it is not known, it is likely bad

Building the system

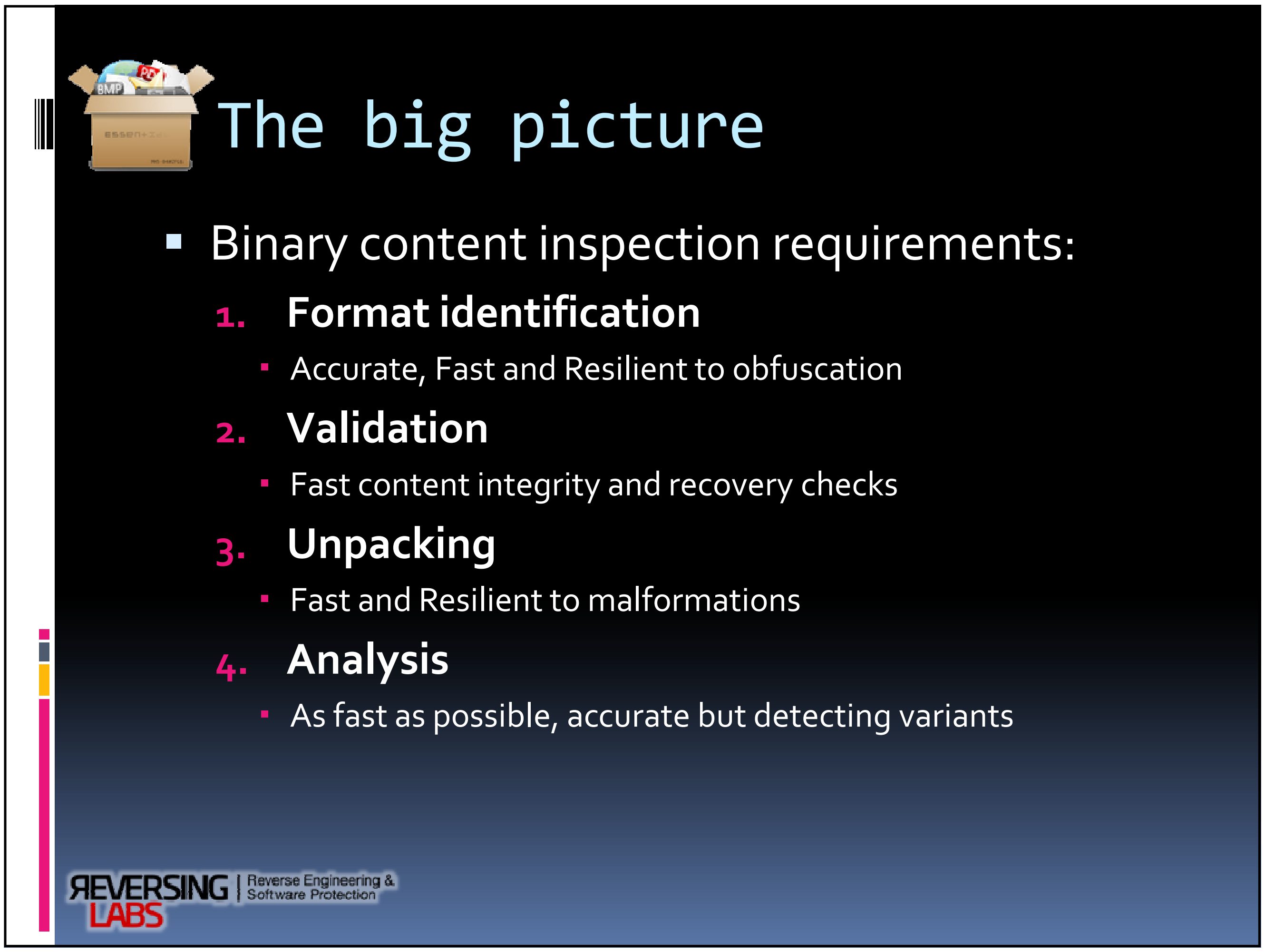
REVERSING LABS | Reverse Engineering & Software Protection



The big picture

- Viable compressed binary content data
  - Documents (PDF, SWF, …)
  - Archives (ZIP, RAR, CAB, …)
  - Archive SFX modules (ZIP, RAR, CAB, …)
  - Installers (InstallShield, InnoSetup, NullSoft, …)
  - Portable executable modifiers (UPX, ASPack, …)
    - Crypters, Packers, Protectors, Bundlers, Hybrids, …

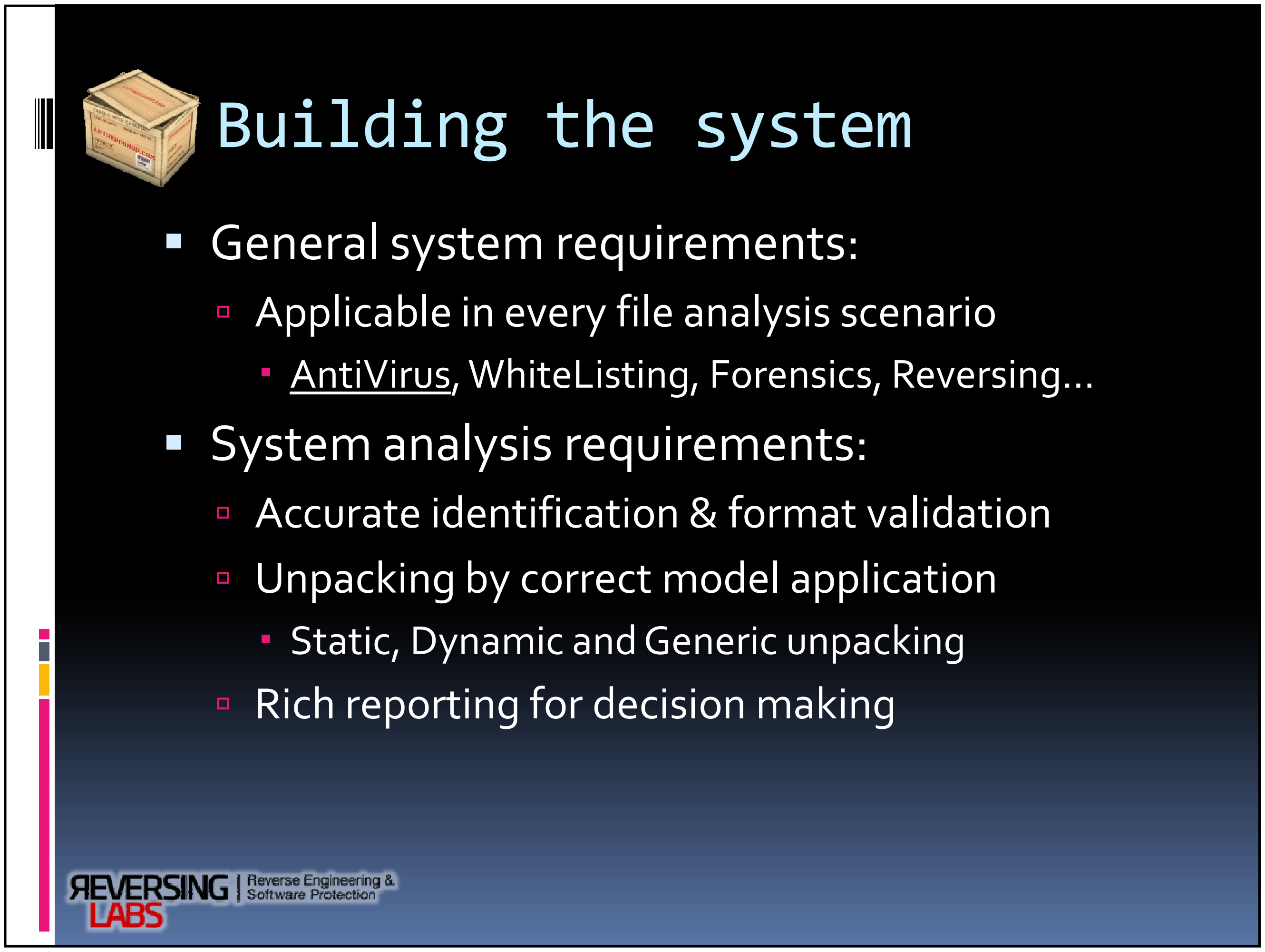
REVERSING LABS | Reverse Engineering & Software Protection



The big picture

- Binary content inspection requirements:
  1. **Format identification**
     - Accurate, Fast and Resilient to obfuscation
  2. **Validation**
     - Fast content integrity and recovery checks
  3. **Unpacking**
     - Fast and Resilient to malformations
  4. **Analysis**
     - As fast as possible, accurate but detecting variants

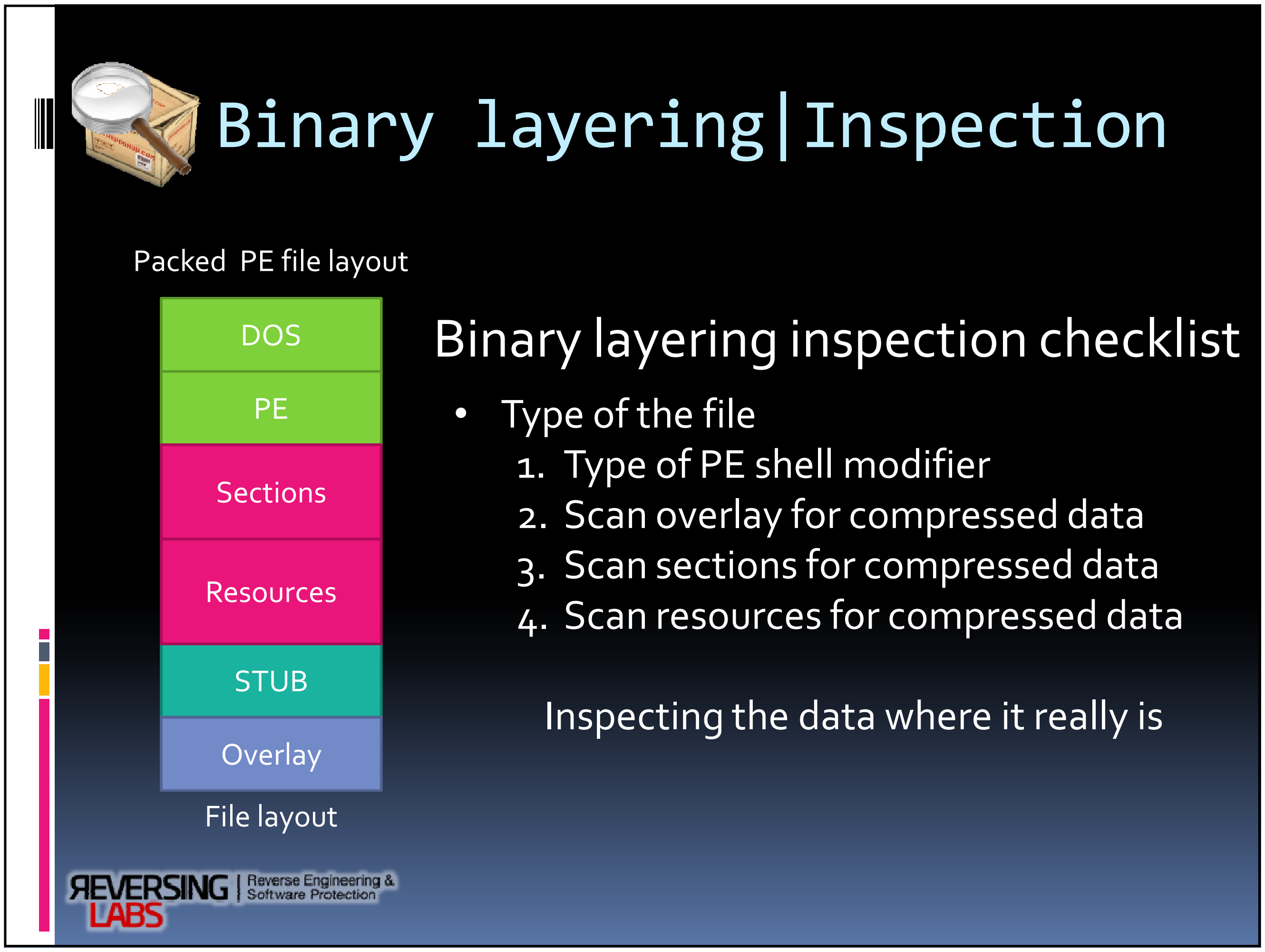
REVERSING LABS | Reverse Engineering & Software Protection



Building the system

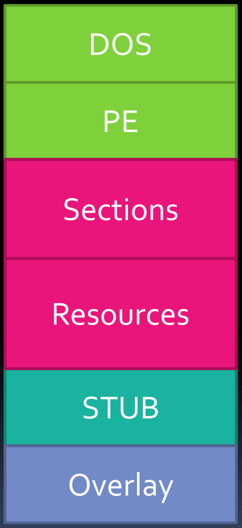
- General system requirements:
  - Applicable in every file analysis scenario
    - AntiVirus, WhiteListing, Forensics, Reversing…
- System analysis requirements:
  - Accurate identification & format validation
  - Unpacking by correct model application
    - Static, Dynamic and Generic unpacking
  - Rich reporting for decision making

REVERSING LABS | Reverse Engineering & Software Protection



Binary layering|Inspection

Packed PE file layout



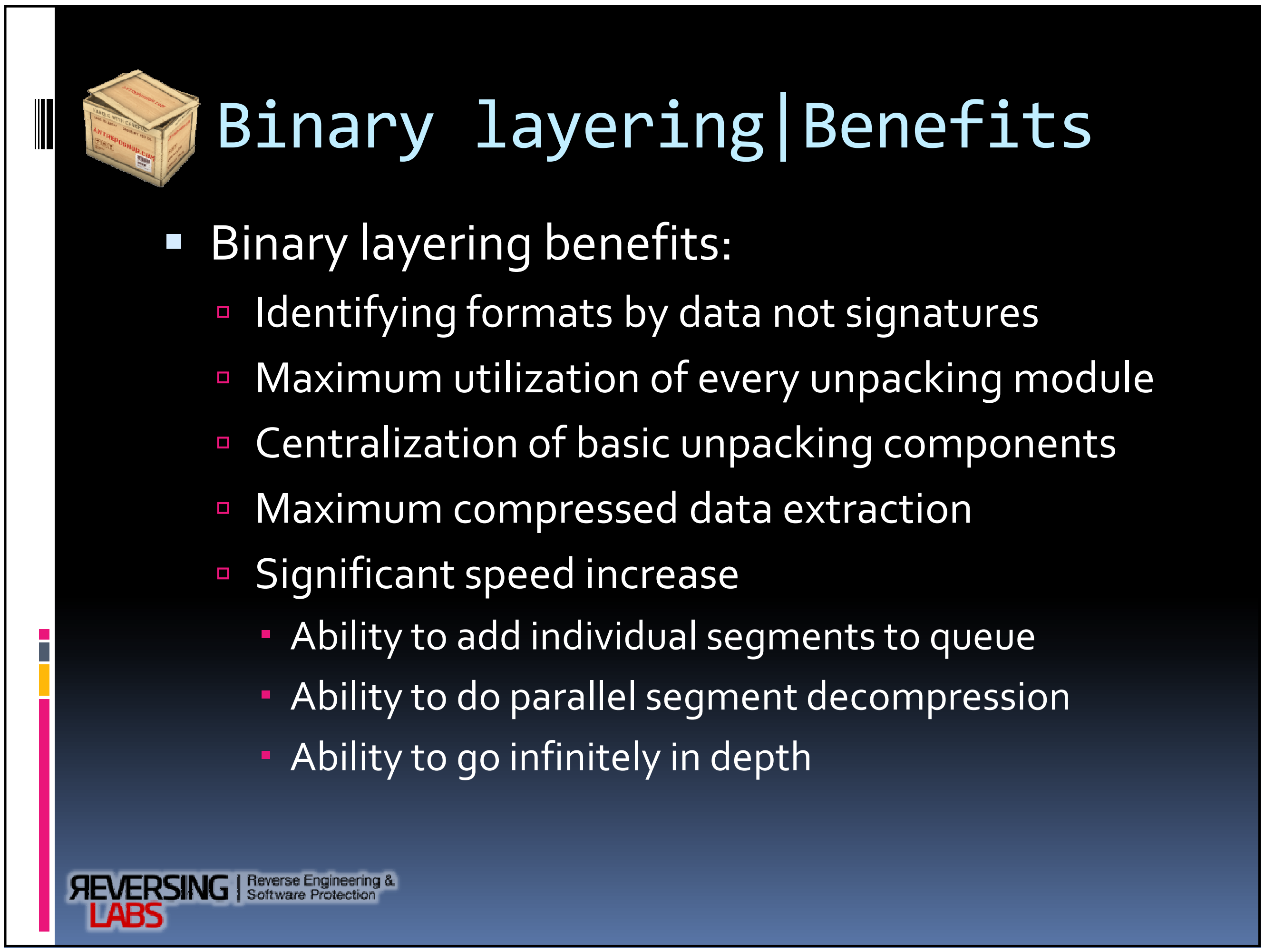
File layout

Binary layering inspection checklist

- Type of the file
  1. Type of PE shell modifier
  2. Scan overlay for compressed data
  3. Scan sections for compressed data
  4. Scan resources for compressed data

Inspecting the data where it really is

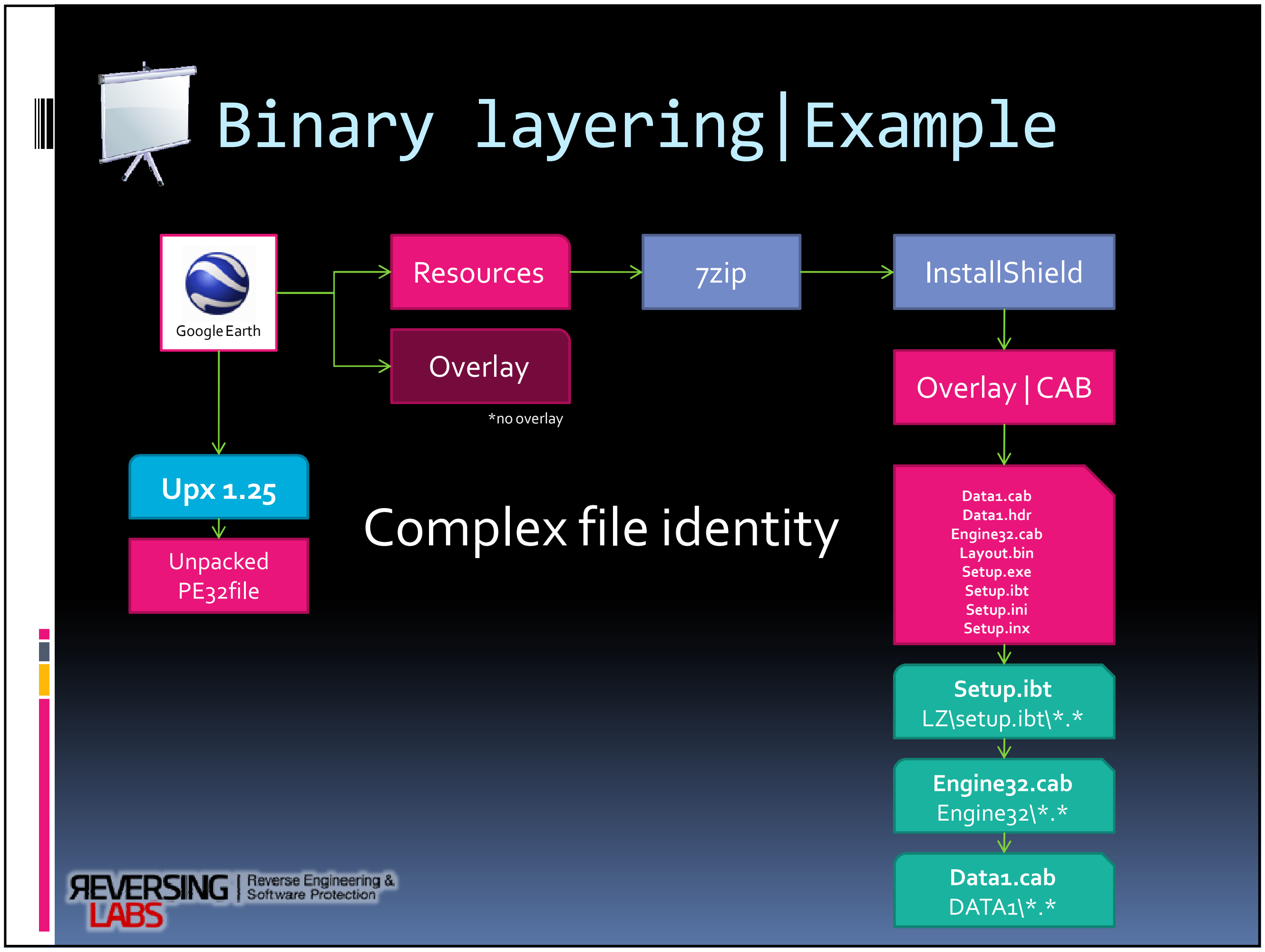
REVERSING LABS | Reverse Engineering & Software Protection



Binary layering|Benefits

- Binary layering benefits:
  - Identifying formats by data not signatures
  - Maximum utilization of every unpacking module
  - Centralization of basic unpacking components
  - Maximum compressed data extraction
  - Significant speed increase
    - Ability to add individual segments to queue
    - Ability to do parallel segment decompression
    - Ability to go infinitely in depth

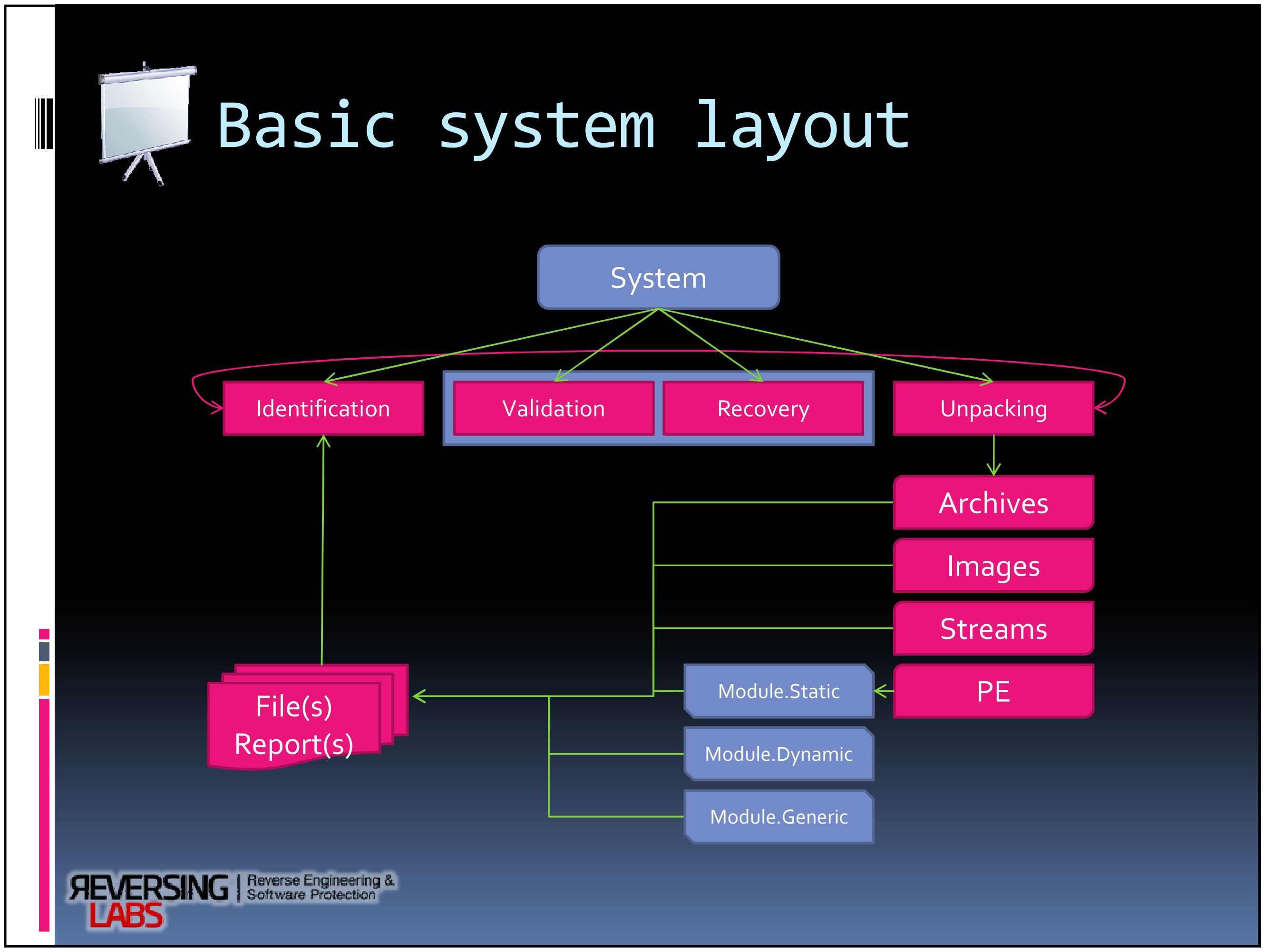
REVERSING LABS | Reverse Engineering & Software Protection



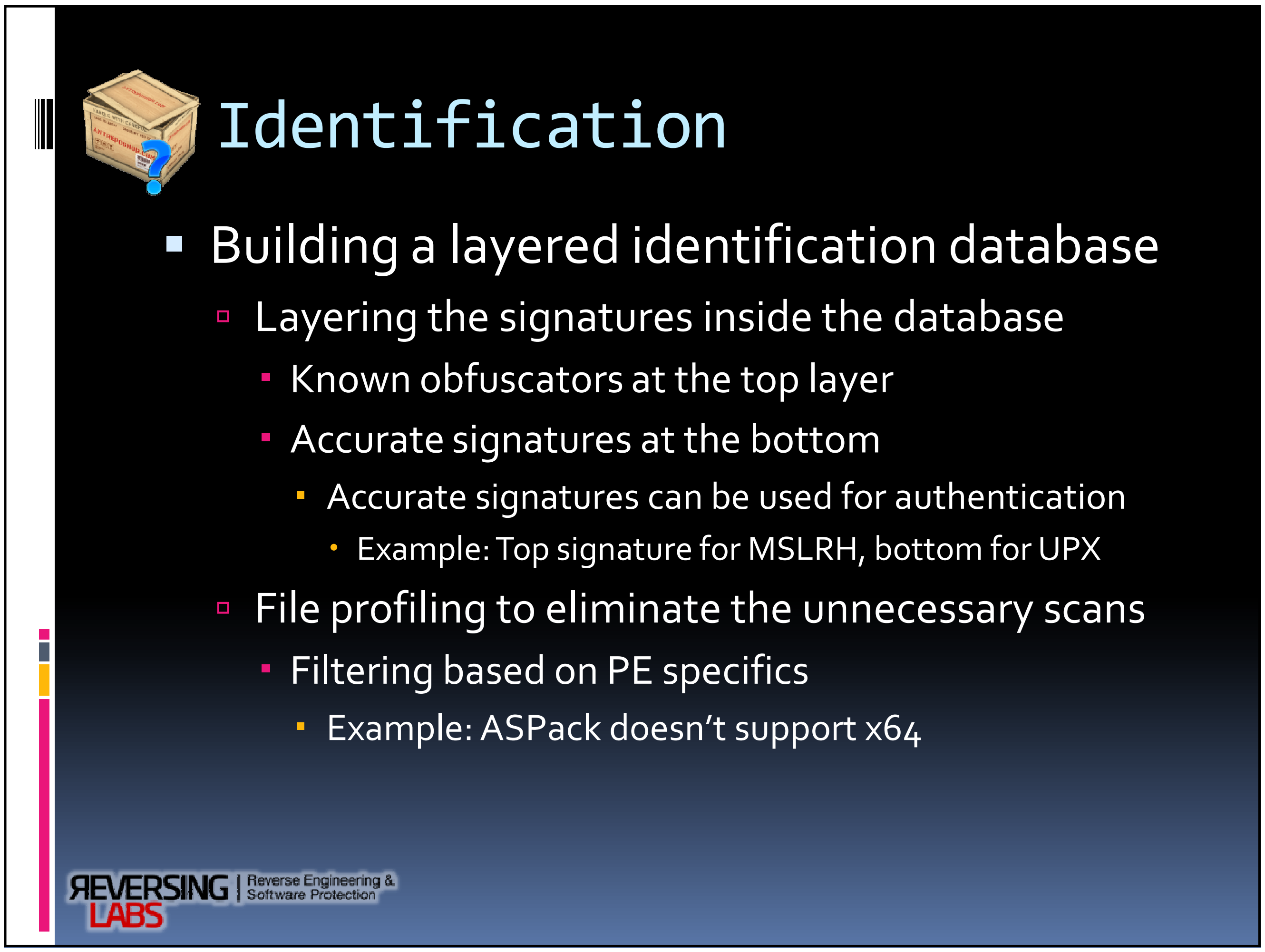
Binary layering|Example

Complex file identity

REVERSING LABS | Reverse Engineering & Software Protection



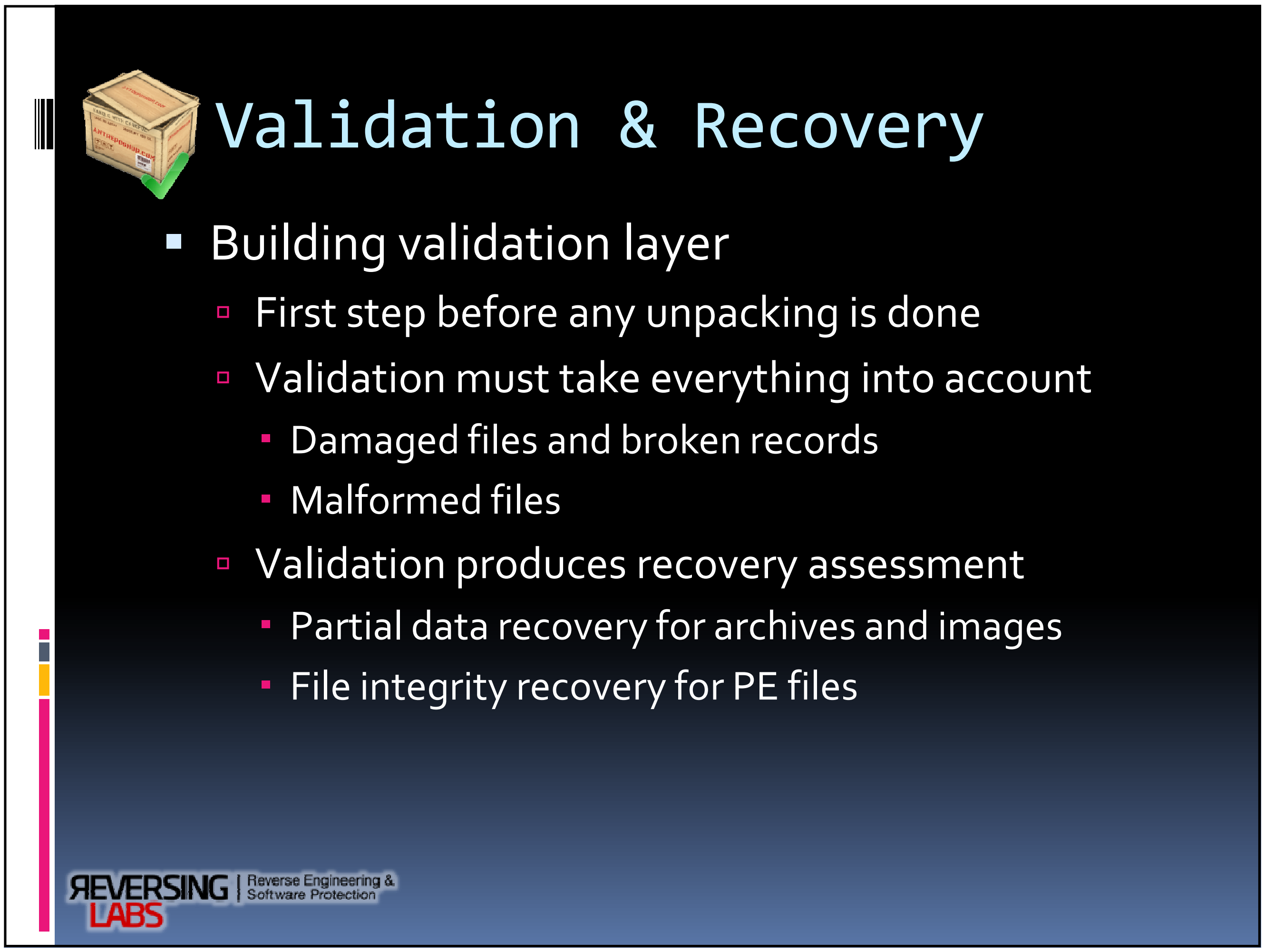
Basic system layout



Identification

- Building a layered identification database
  - Layering the signatures inside the database
    - Known obfuscators at the top layer
    - Accurate signatures at the bottom
      - Accurate signatures can be used for authentication
        - Example: Top signature for MSLRH, bottom for UPX
  - File profiling to eliminate the unnecessary scans
    - Filtering based on PE specifics
      - Example: ASPack doesn't support x64

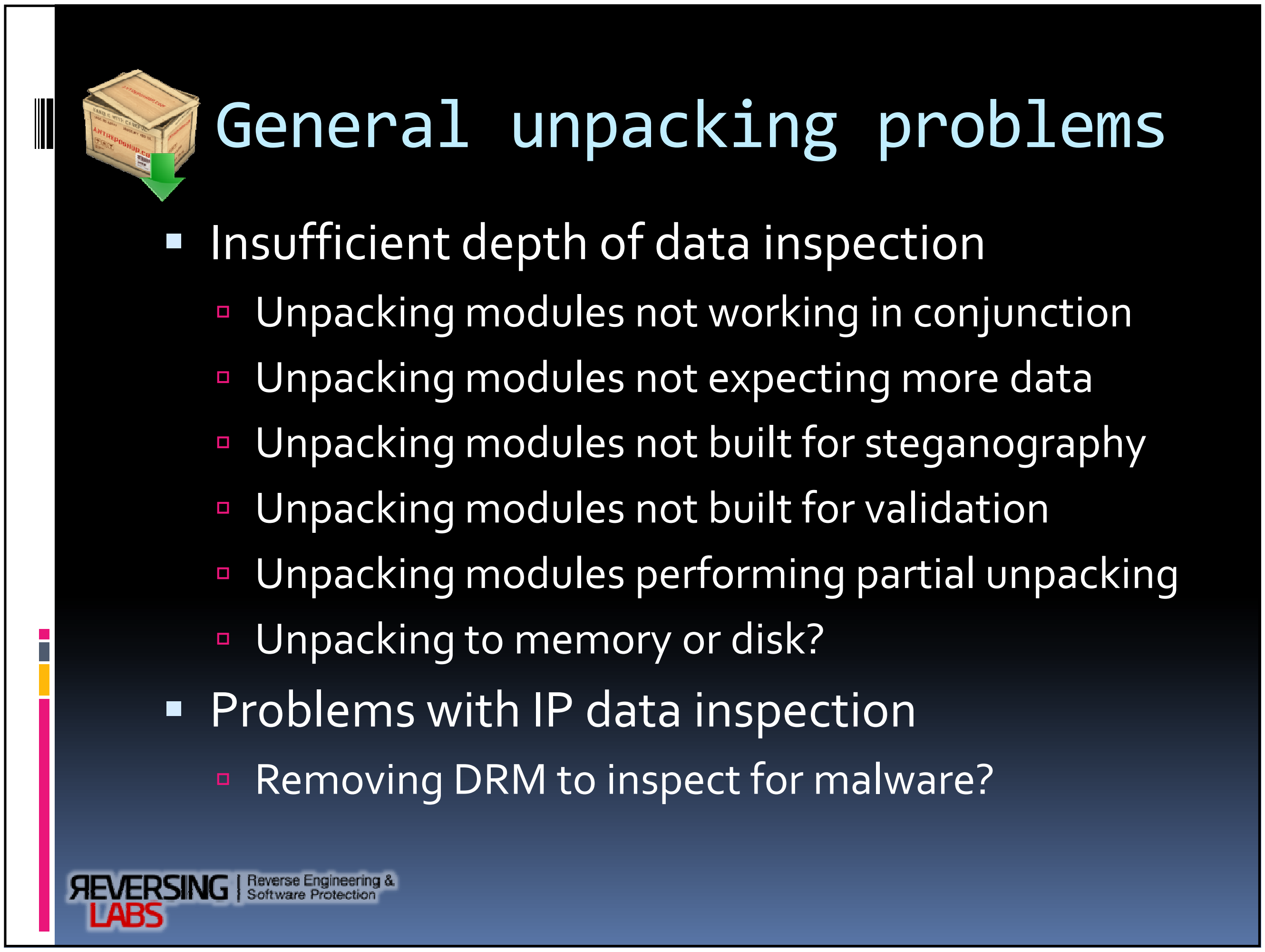
REVERSING LABS | Reverse Engineering & Software Protection



Validation & Recovery

- Building validation layer
  - First step before any unpacking is done
  - Validation must take everything into account
    - Damaged files and broken records
    - Malformed files
  - Validation produces recovery assessment
    - Partial data recovery for archives and images
    - File integrity recovery for PE files

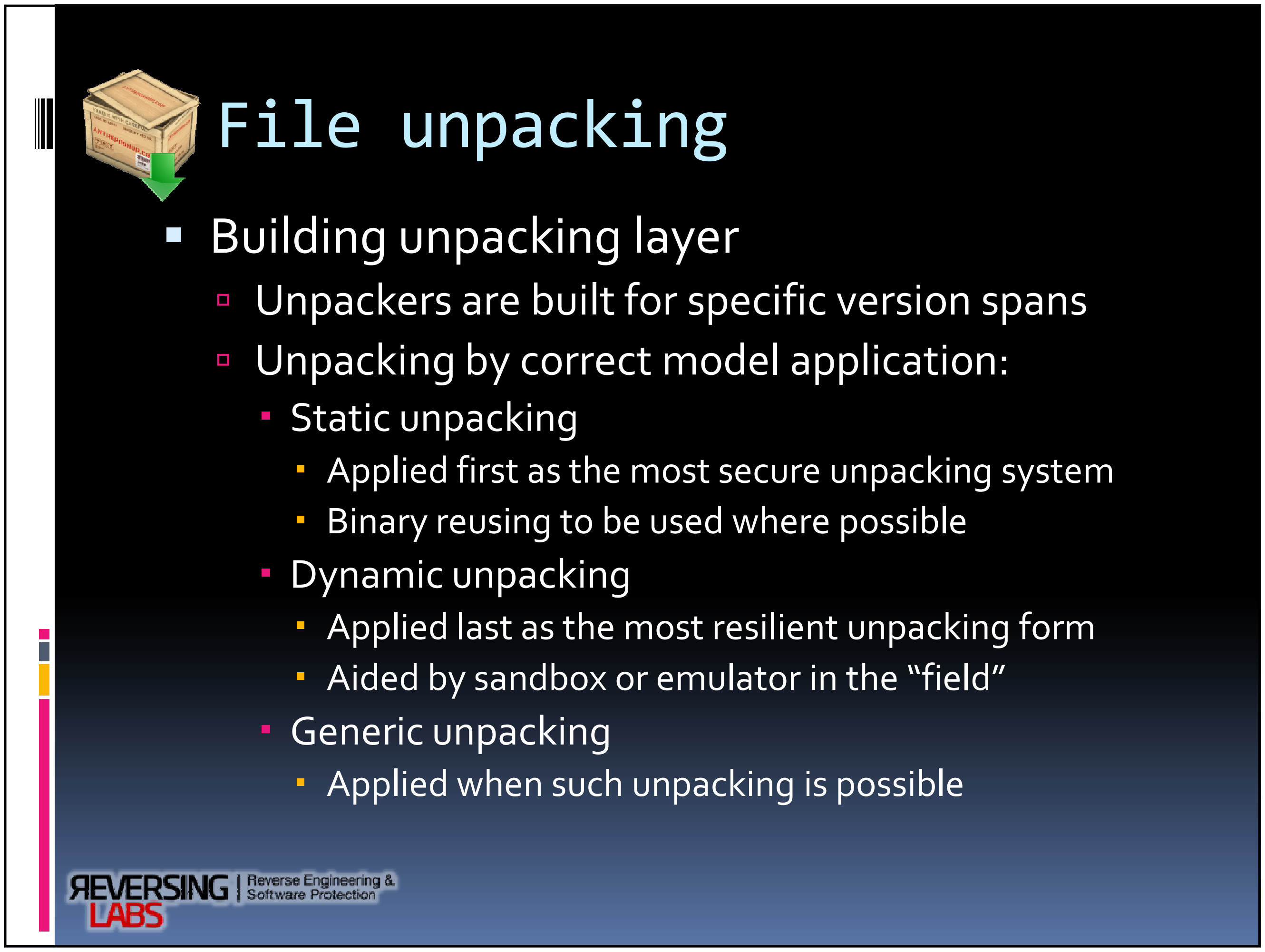
REVERSING LABS | Reverse Engineering & Software Protection



General unpacking problems

- Insufficient depth of data inspection
  - Unpacking modules not working in conjunction
  - Unpacking modules not expecting more data
  - Unpacking modules not built for steganography
  - Unpacking modules not built for validation
  - Unpacking modules performing partial unpacking
  - Unpacking to memory or disk?
- Problems with IP data inspection
  - Removing DRM to inspect for malware?

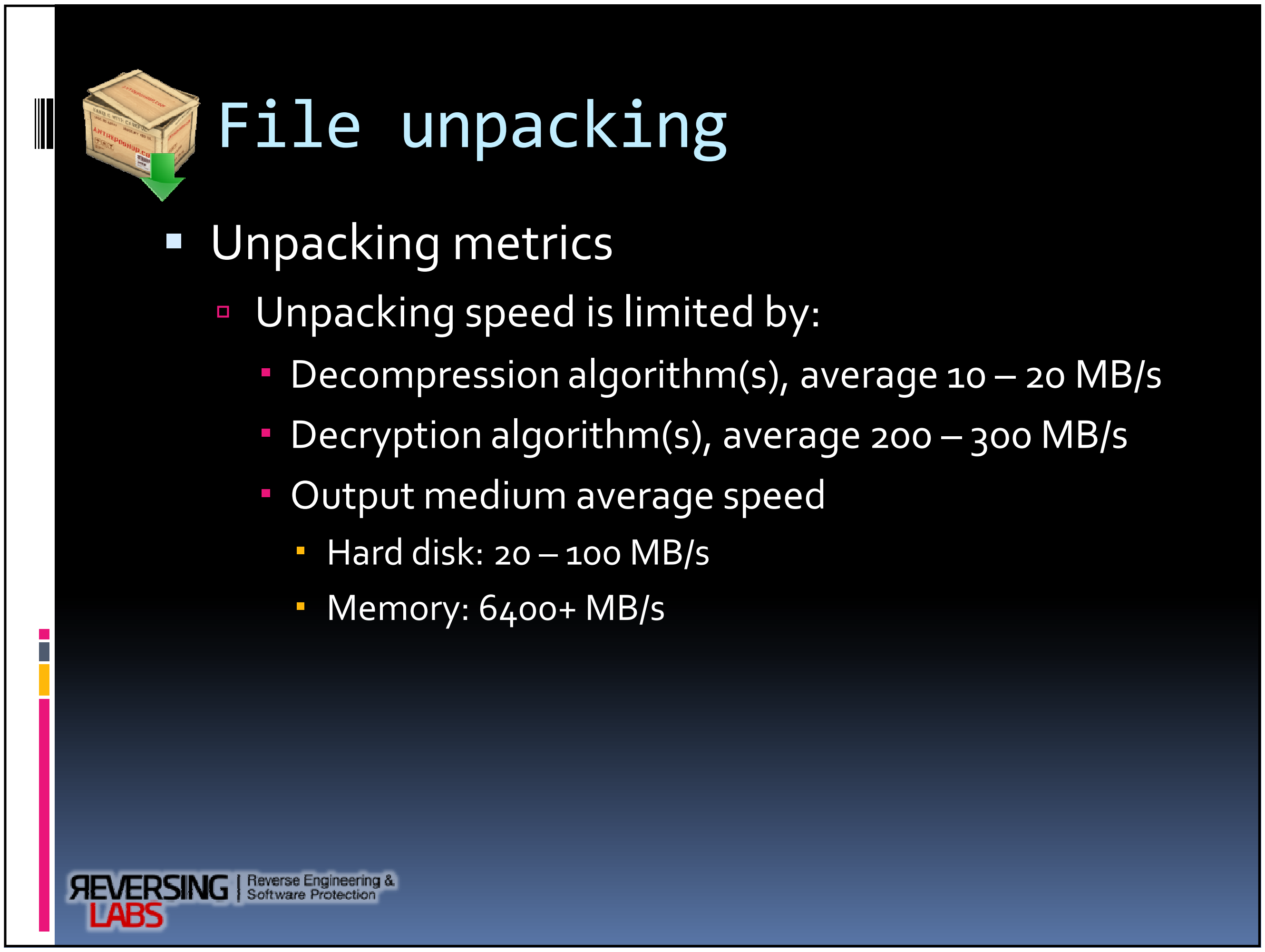
REVERSING LABS | Reverse Engineering & Software Protection



File unpacking

- Building unpacking layer
  - Unpackers are built for specific version spans
  - Unpacking by correct model application:
    - Static unpacking
      - Applied first as the most secure unpacking system
      - Binary reusing to be used where possible
    - Dynamic unpacking
      - Applied last as the most resilient unpacking form
      - Aided by sandbox or emulator in the “field”
    - Generic unpacking
      - Applied when such unpacking is possible

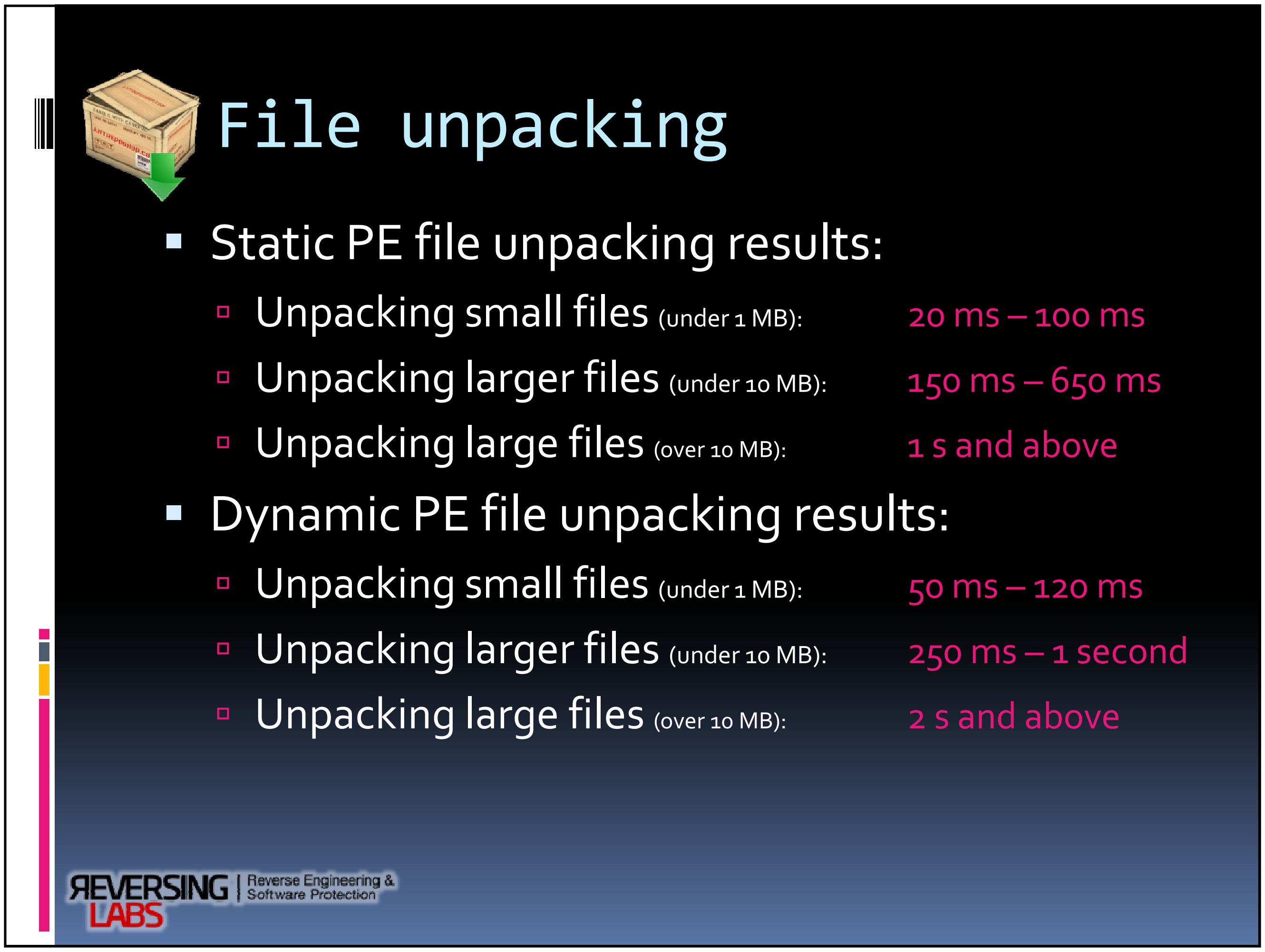
REVERSING LABS | Reverse Engineering & Software Protection



File unpacking

- Unpacking metrics
  - Unpacking speed is limited by:
    - Decompression algorithm(s), average 10 – 20 MB/s
    - Decryption algorithm(s), average 200 – 300 MB/s
    - Output medium average speed
      - Hard disk: 20 – 100 MB/s
      - Memory: 6400+ MB/s

REVERSING LABS | Reverse Engineering & Software Protection



File unpacking

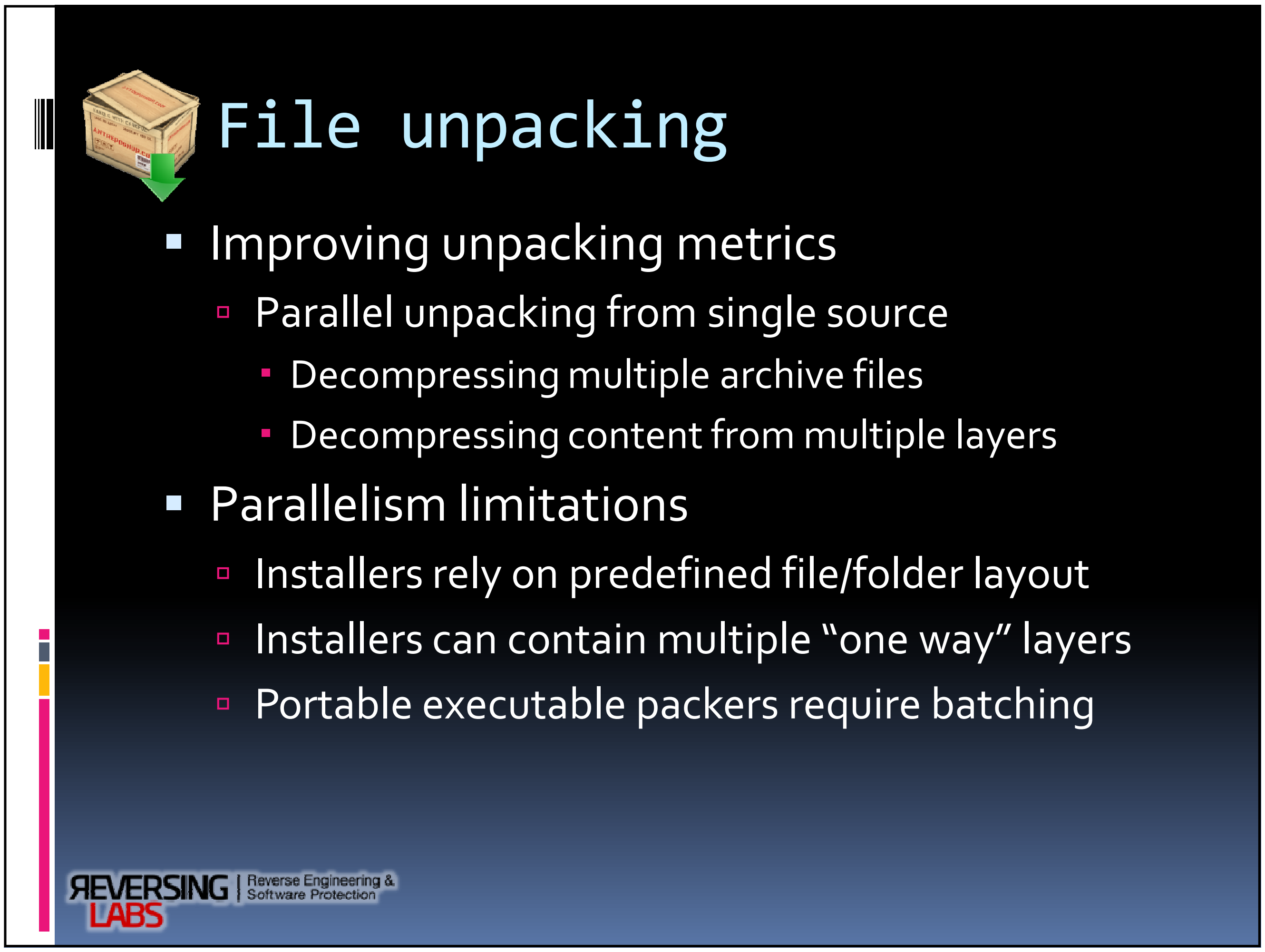
- Static PE file unpacking results:

Unpacking small files (under 1 MB):	20 ms – 100 ms
Unpacking larger files (under 10 MB):	150 ms – 650 ms
Unpacking large files (over 10 MB):	1 s and above

- Dynamic PE file unpacking results:

Unpacking small files (under 1 MB):	50 ms – 120 ms
Unpacking larger files (under 10 MB):	250 ms – 1 second
Unpacking large files (over 10 MB):	2 s and above

REVERSING LABS | Reverse Engineering & Software Protection

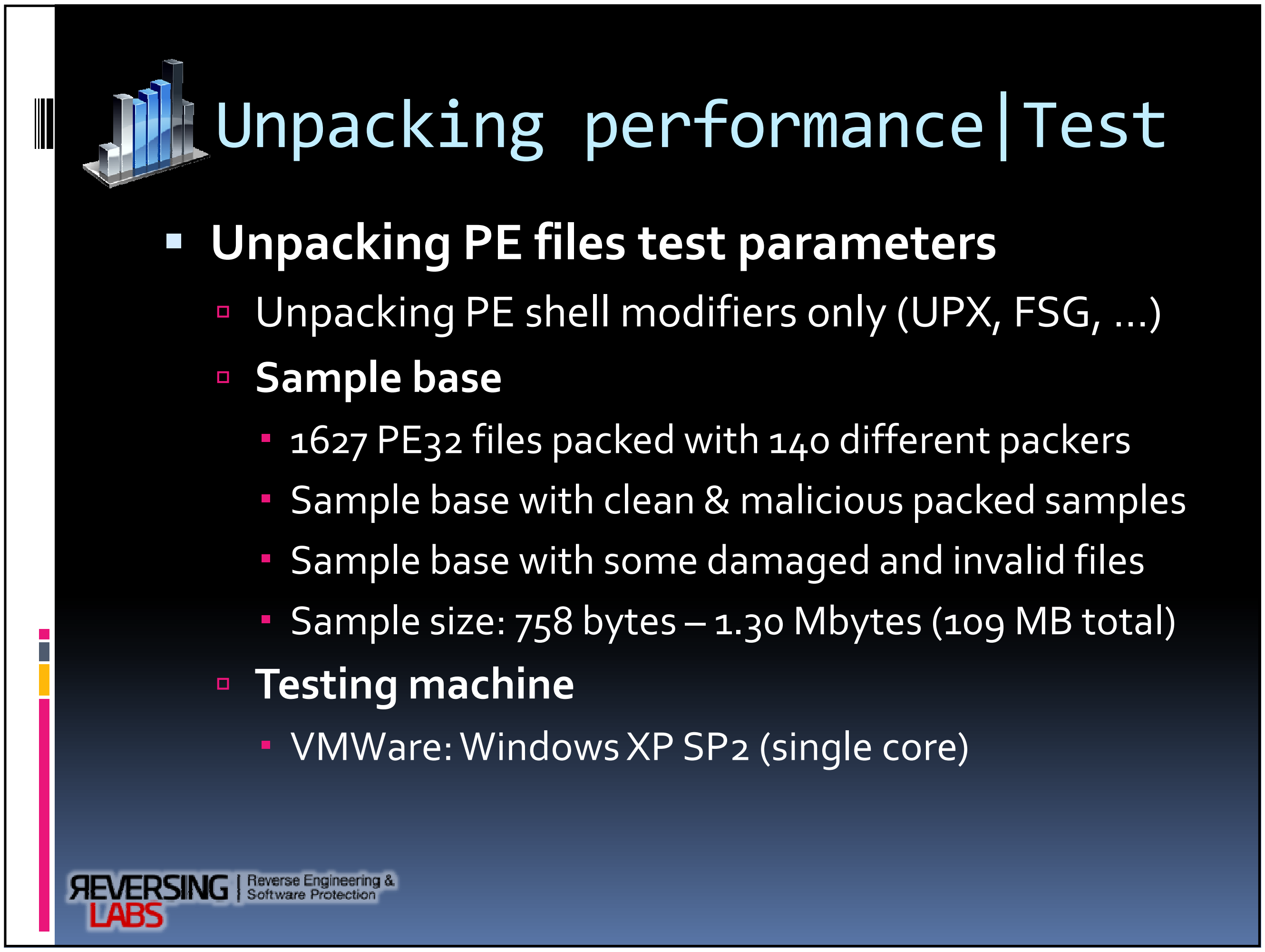


File unpacking

- Improving unpacking metrics
  - Parallel unpacking from single source
    - Decompressing multiple archive files
    - Decompressing content from multiple layers
- Parallelism limitations
  - Installers rely on predefined file/folder layout
  - Installers can contain multiple “one way” layers
  - Portable executable packers require batching

ЯEVERSING LABS | Reverse Engineering & Software Protection

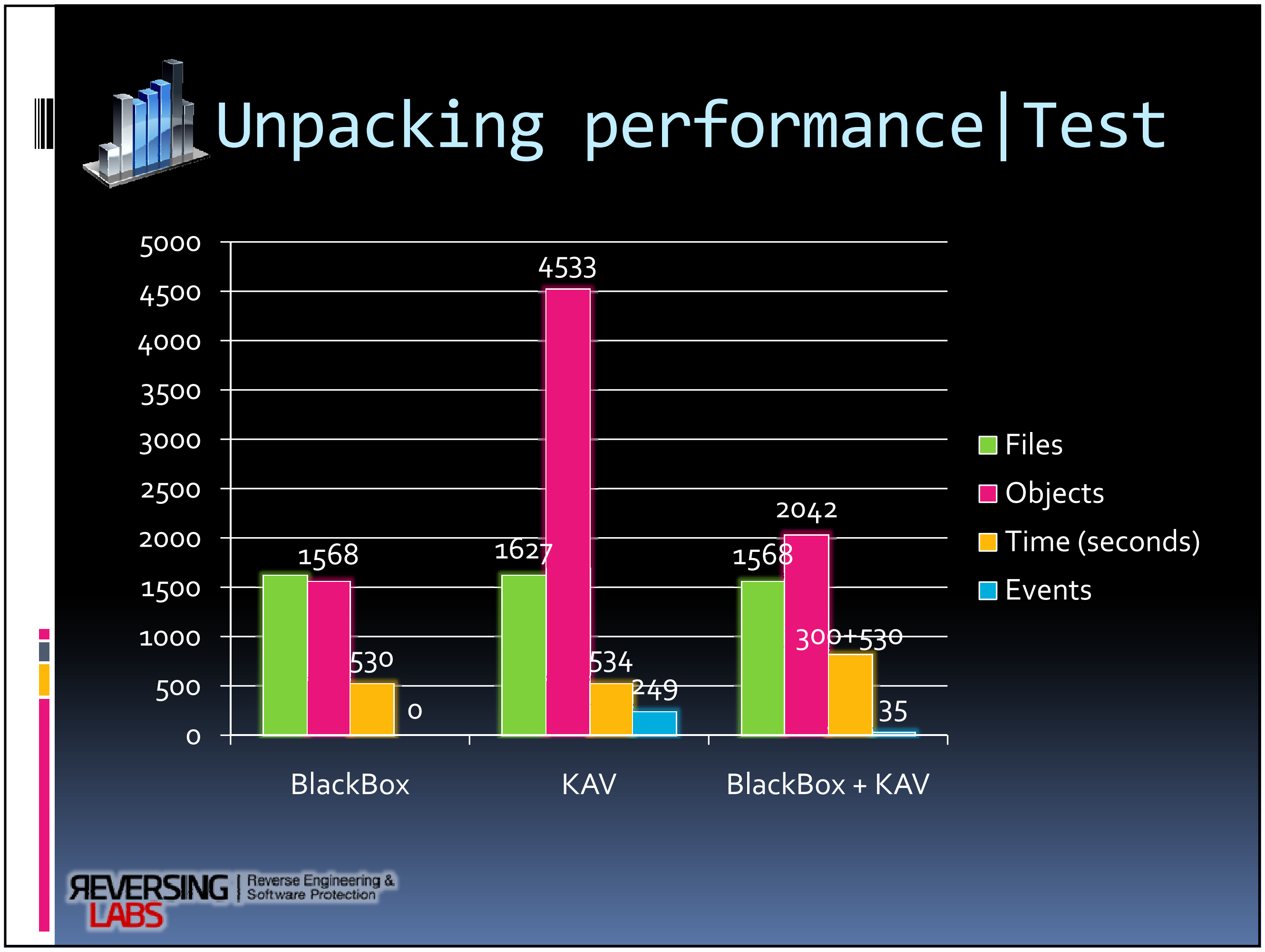
Performance testing



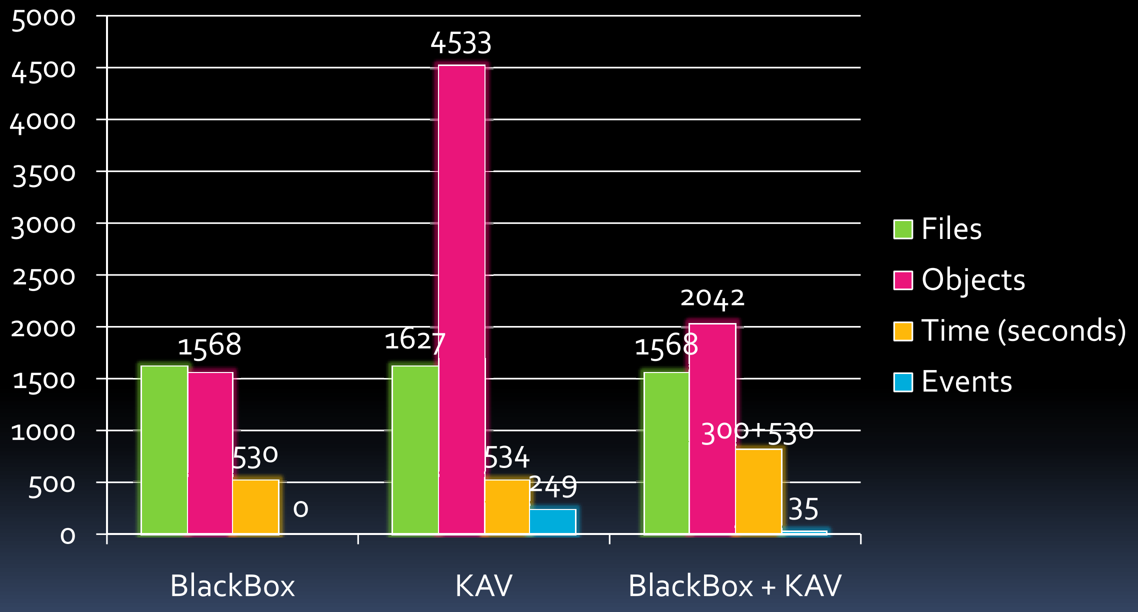
Unpacking performance|Test

- **Unpacking PE files test parameters**
  - Unpacking PE shell modifiers only (UPX, FSG, …)
  - **Sample base**
    - 1627 PE32 files packed with 140 different packers
    - Sample base with clean & malicious packed samples
    - Sample base with some damaged and invalid files
    - Sample size: 758 bytes – 1.30 Mbytes (109 MB total)
  - **Testing machine**
    - VMWare: Windows XP SP2 (single core)

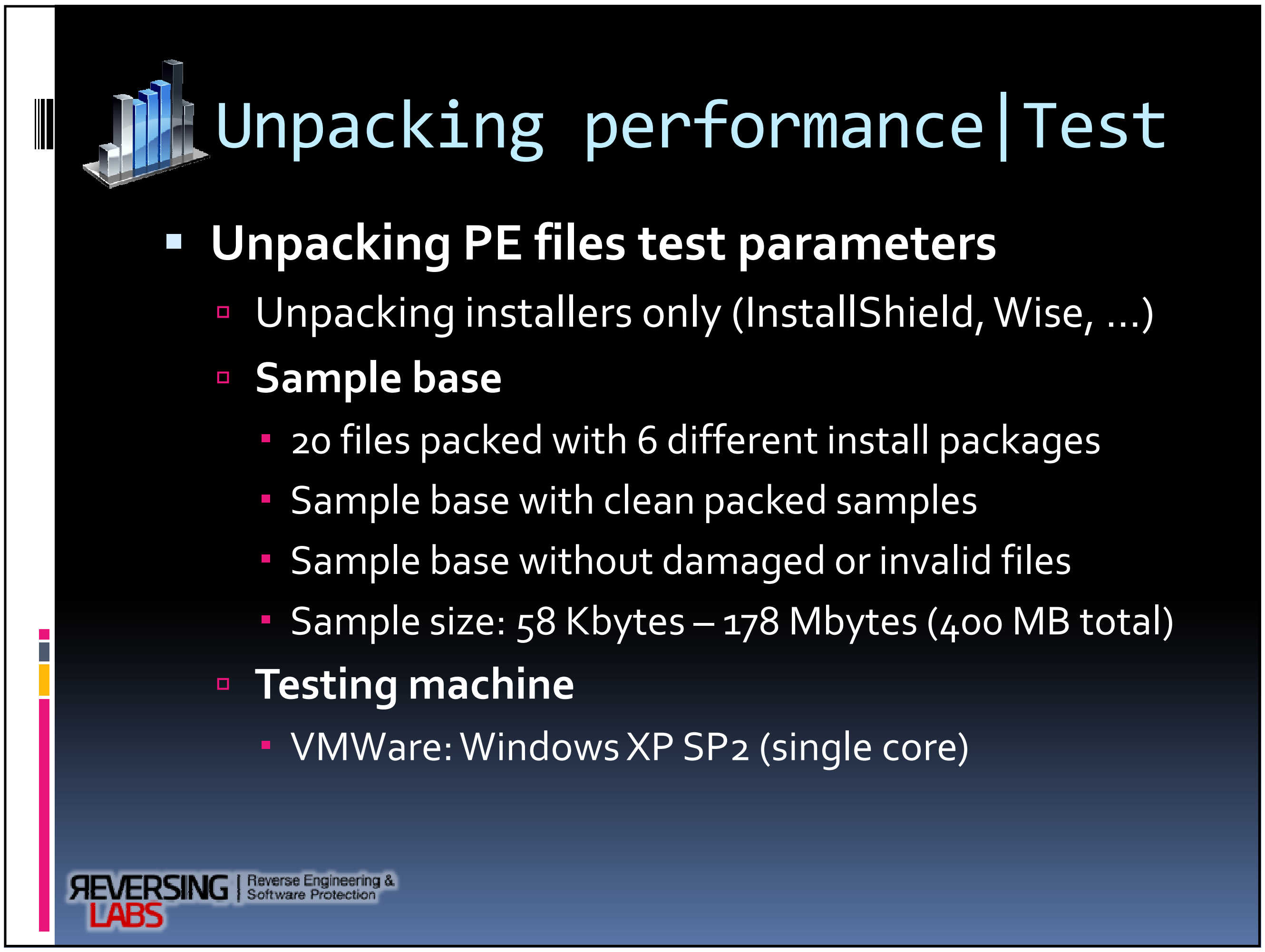
REVERSING LABS | Reverse Engineering & Software Protection



Unpacking performance|Test



REVERSING LABS | Reverse Engineering & Software Protection

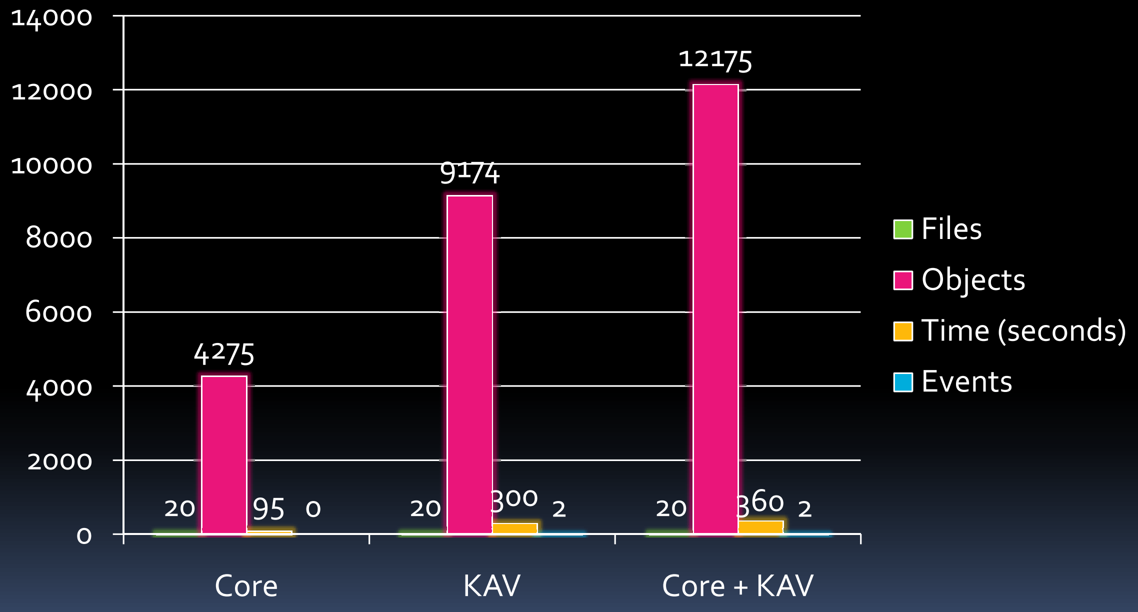


Unpacking performance|Test

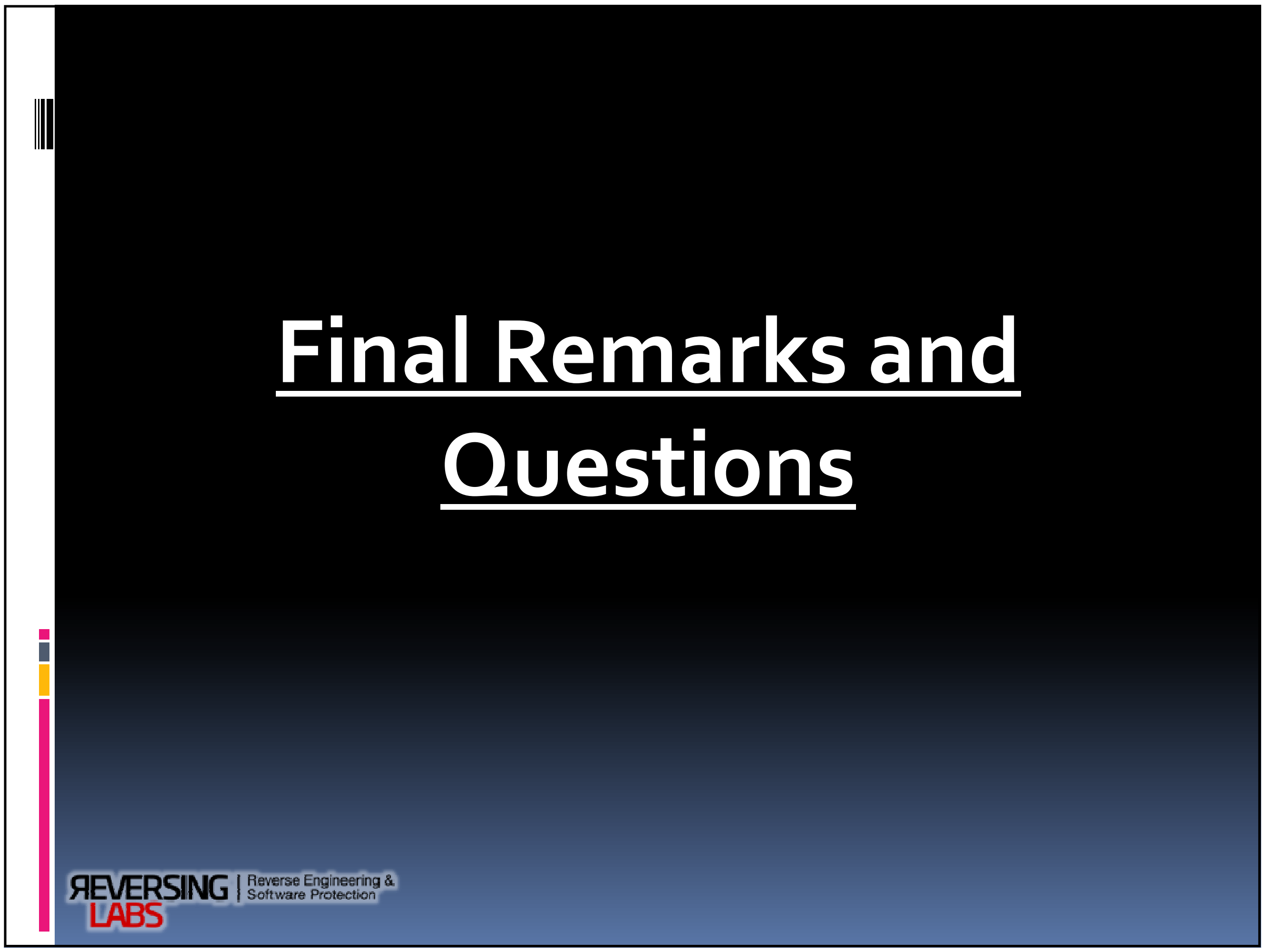
- **Unpacking PE files test parameters**
  - Unpacking installers only (InstallShield, Wise, ...)
  - **Sample base**
    - 20 files packed with 6 different install packages
    - Sample base with clean packed samples
    - Sample base without damaged or invalid files
    - Sample size: 58 Kbytes – 178 Mbytes (400 MB total)
  - **Testing machine**
    - VMWare: Windows XP SP2 (single core)

REVERSING LABS | Reverse Engineering & Software Protection

Unpacking performance|Test



REVERSING LABS | Reverse Engineering & Software Protection



Final Remarks and Questions

REVERSING LABS | Reverse Engineering & Software Protection