PIXELHEIST

PULLING FILES OUT FROM YOUR VIRTUAL DESKTOP SCREEN.

JEREMY SOH

@BREAKTOPROTECT

24 SEP 2019

ABOUT ME

JS @BREAKTOPROTECT

BLOGS AT BREAKTOPROTECT.NET

CODE AT GITHUB.COM/BREAKTOPROTECT

OFFENSIVE TACTICS & RESEARCH & ENGINEERING

DISCLAIMER!

All views expressed are of my own and do not represent the opinions of **any entity whatsoever** with which I have been, am now or will be affiliated.

THE ‘KILL CHAIN’

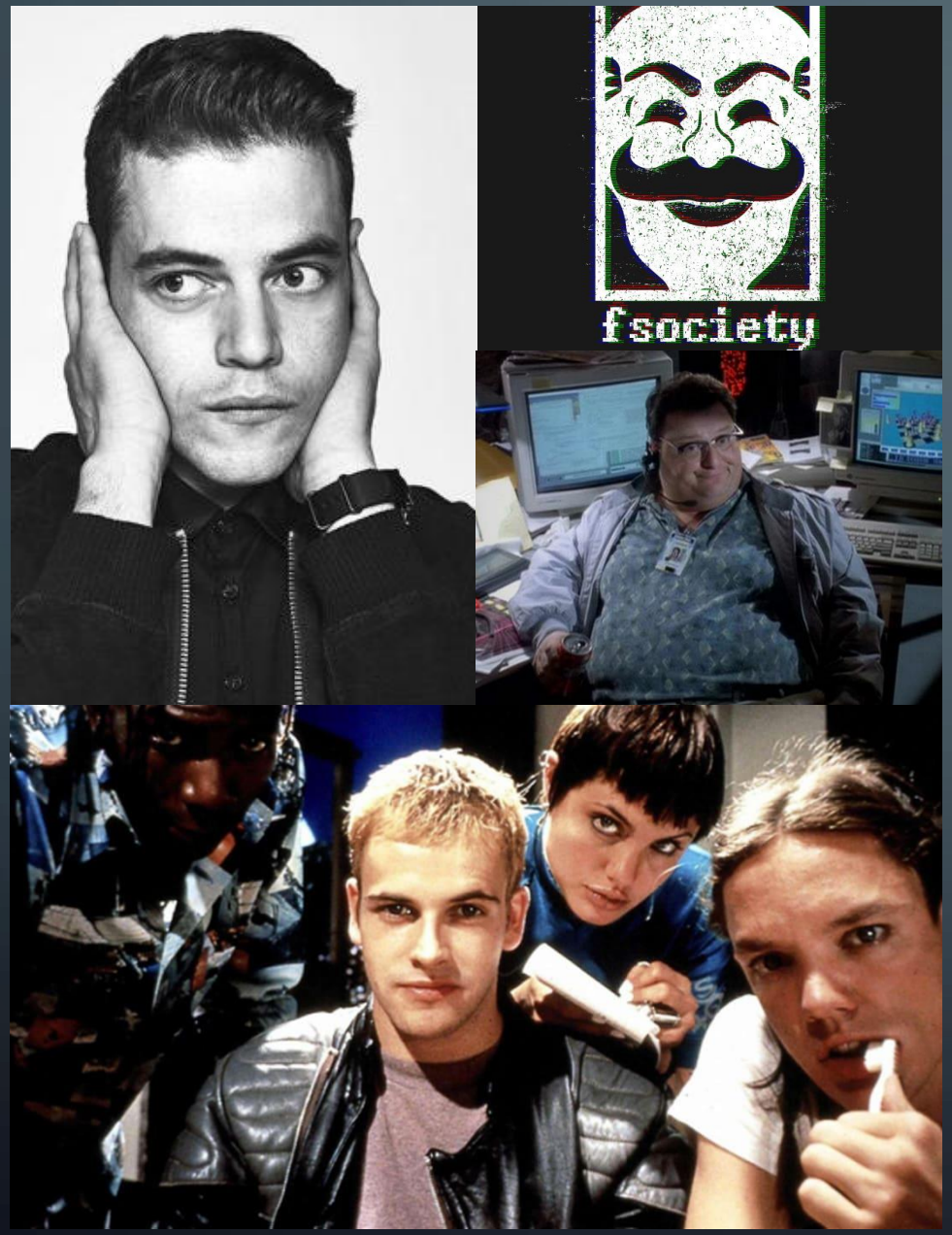
MITRE'S ATT&CK

ATT&CK Matrix for Enterprise

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Data Destruction
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Encrypted for Impact
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Connection Proxy	Data Encrypted	Defacement
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Disk Content Wipe
Replication Through Removable Media	Control Panel Items	AppInit DLLs	Application Shimming	Clear Command History	Credentials in Files	File and Directory Discovery	Logon Scripts	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Structure Wipe

- Exfiltration
- Automated Exfiltration
- Data Compressed
- Data Encrypted
- Data Transfer Size Limits
- Exfiltration Over Alternative Protocol

‘ACTIONS ON OBJECTIVES’



DATA THEFT - POV

- Objective: To bring acquired data out of secure environment
- Challenges:
  - To evade detection;
  - To bypass multiple controls;
  - To leave little to no evidence.
- Rate the volume of data proportional to noise / time-required

MODERN DAY ANTI-EXFIL AKA D.L.P.

- Term focuses on Egress than Ingress
- Deals with data classification
- More sensitive the data, less likely it leaves the org
- “Data at-rest, in-use and in-motion”
- May tag team with insider threat / fraud team
- Significant Emphasis on Insider Threats
  - Employees who “backs up” customer information on cloud storage
  - Employees who “shares” information with prospective buyers
  - Employees who “wants to see the company 🔥”

DEFINITION OF D.L.P.

“Data loss prevention software detects potential data breaches/data ex-filtration transmissions and prevents them by monitoring, detecting and blocking sensitive data while in use, in motion, and at rest.”

MODERN DAY DLP PREVENTIVE CONTROLS

CONTROLLED PHYSICAL ACCESS

RESTRICTED STORAGE MEDIA

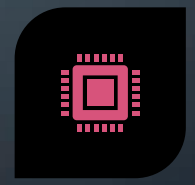
ENCRYPTION

EMAIL FILTERS

WEB RESTRICTION

ENDPOINT PROTECTION

MOBILE DEVICE MANAGEMENT



VIRTUALIZED DESKTOP

BAD GUYS GET CREATIVE ON EXFIL

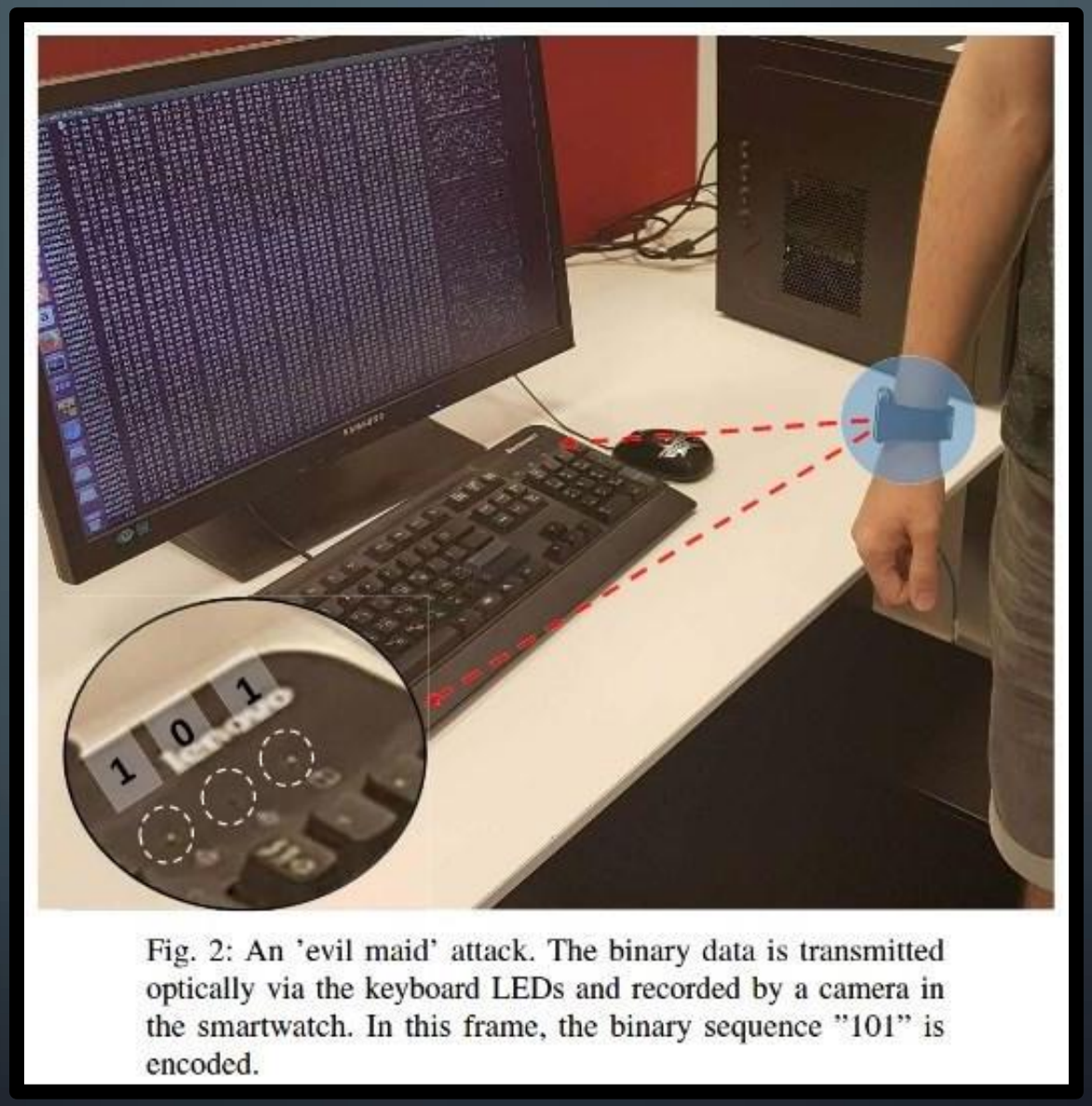
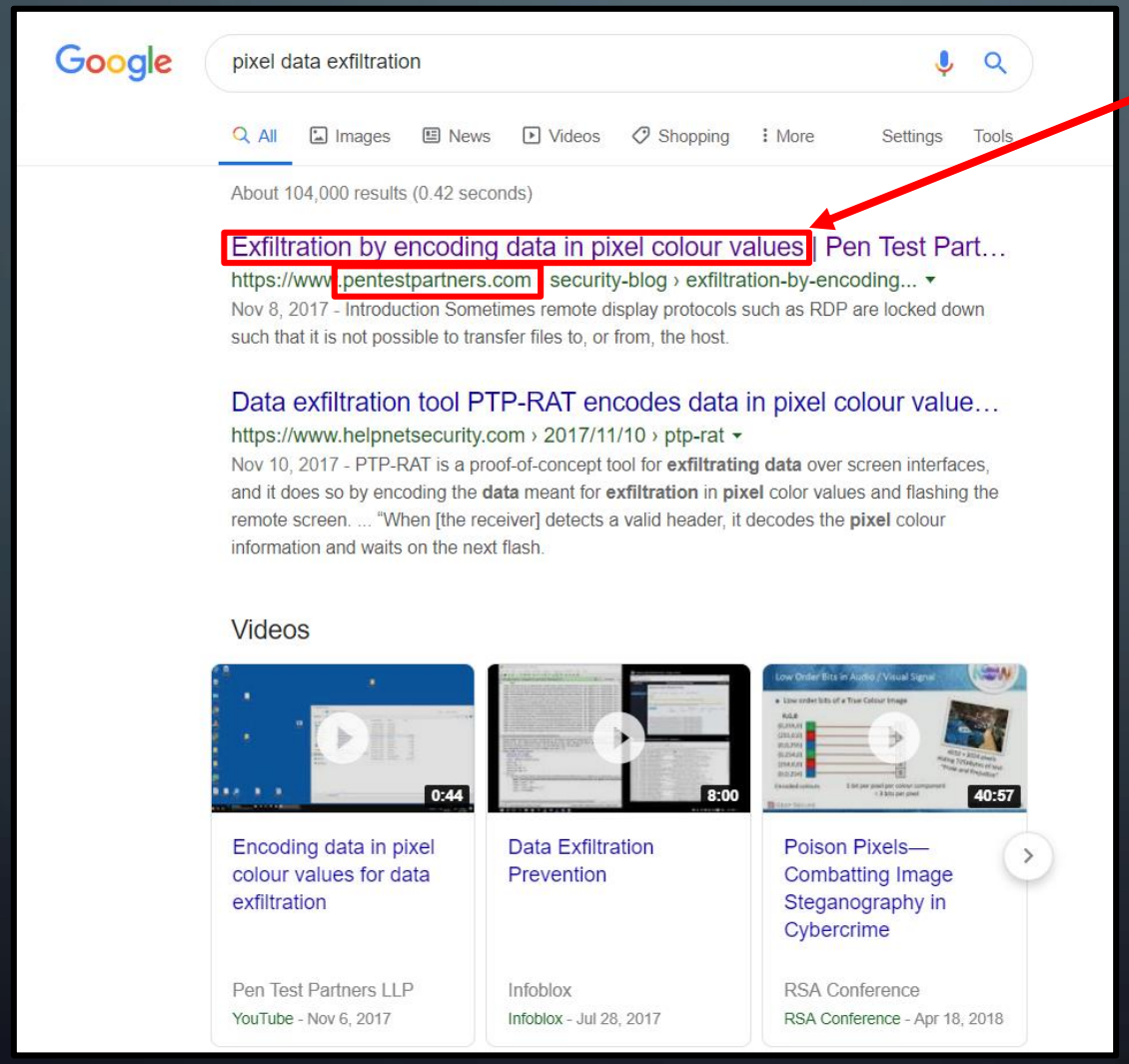


Fig. 2: An 'evil maid' attack. The binary data is transmitted optically via the keyboard LEDs and recorded by a camera in the smartwatch. In this frame, the binary sequence "101" is encoded.

Source: https://www.zdnet.com/article/academics-steal-data-from-air-gapped-systems-via-a-keyboards-leds/

WHAT ABOUT VIA MONITOR SCREEN?

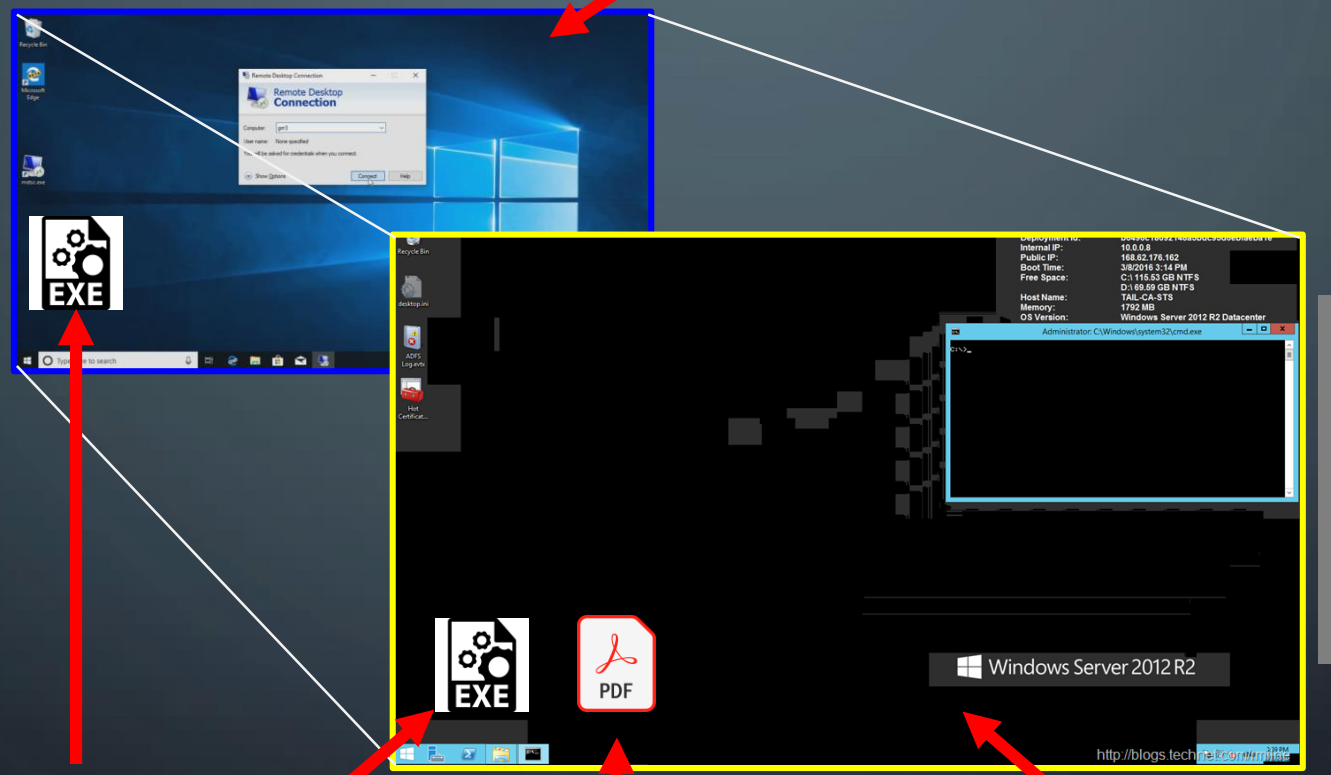


- First Appeared in Nov 2017
- Published by **Pentest Partners**
- Encodes/decodes Pixels for transfer of files

HIGH-LEVEL – HOW DOES IT WORK?

- A single binary with two modes: Sender/Transmitter & Receiver
- Sender/Transmitter mode to be loaded on ‘source’ to encode a file’s content into images to ‘splash on screen’
- Receiver mode to be loaded on ‘destination’ to receive images, decode and restore

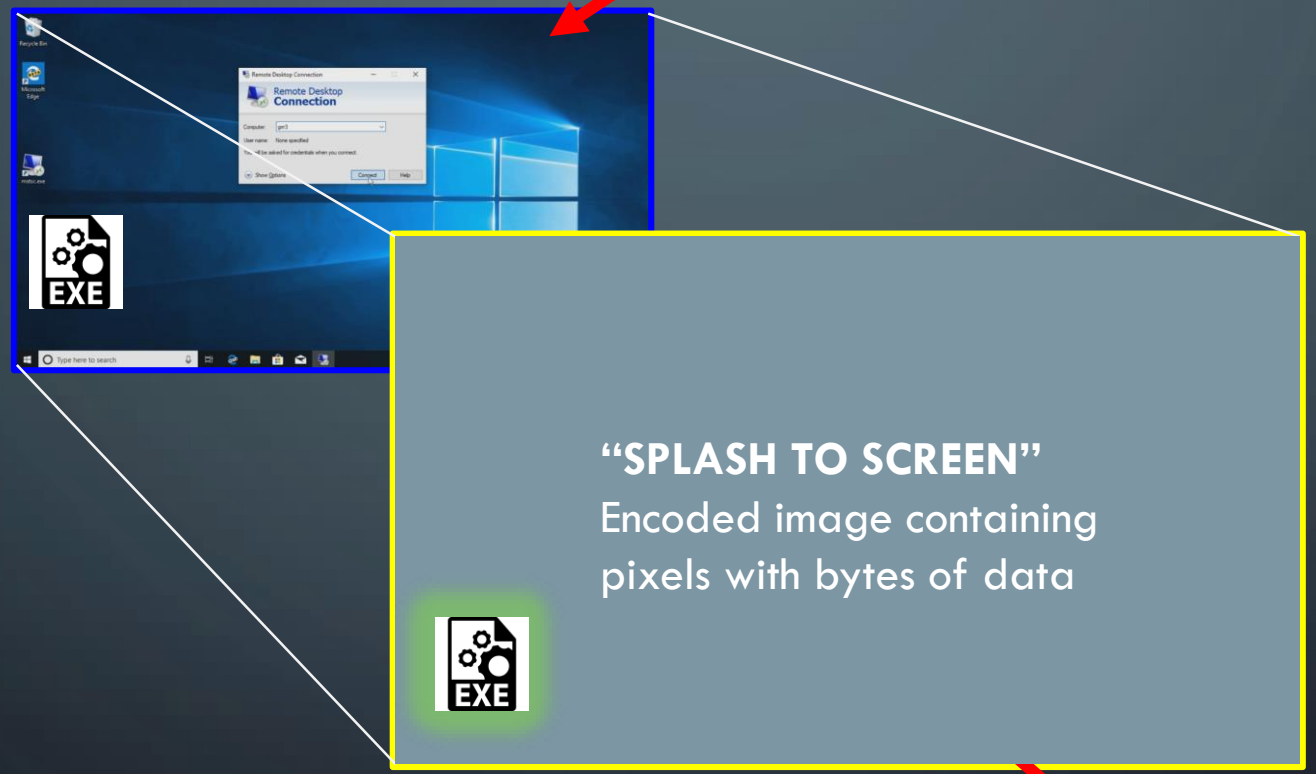
VISUALIZE



Virtual Desktop Interface –
A virtualized machine sitting at the datacentre or cloud!

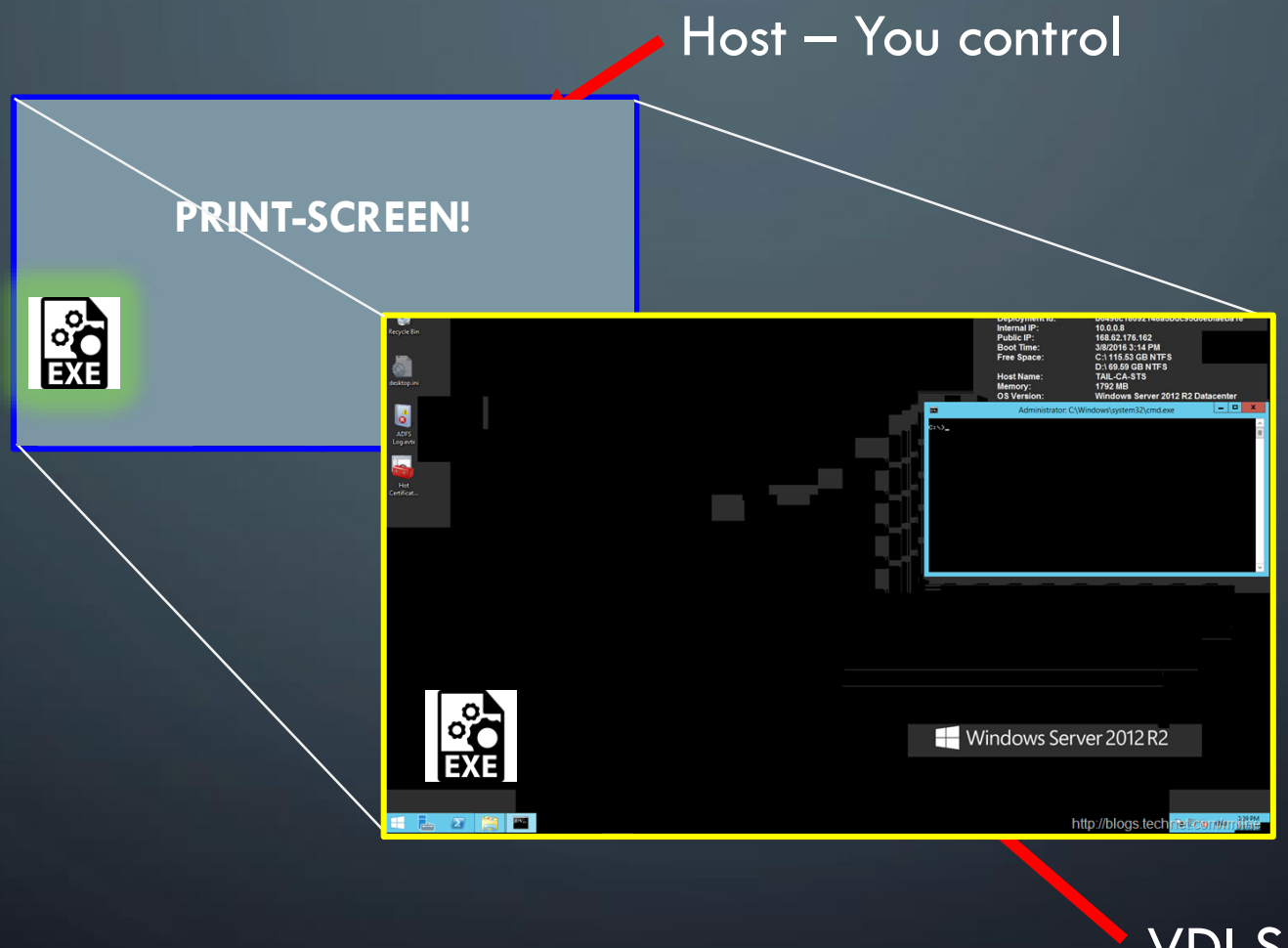
VISUALIZE

Host – You control



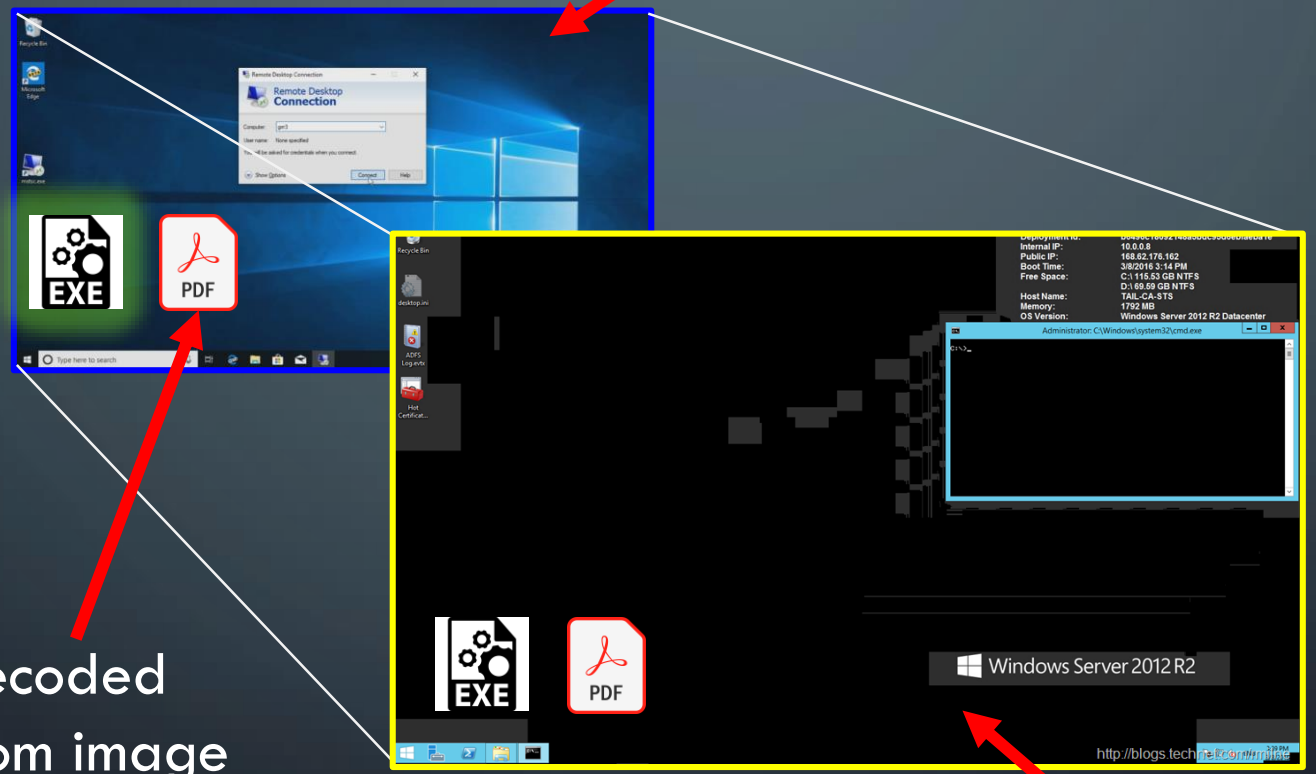
VDI Session –
You have access

VISUALIZE


Host – You control
PRINT-SCREEN!
EXE
Windows Server 2012 R2
VDI Session –
You have access

VISUALIZE

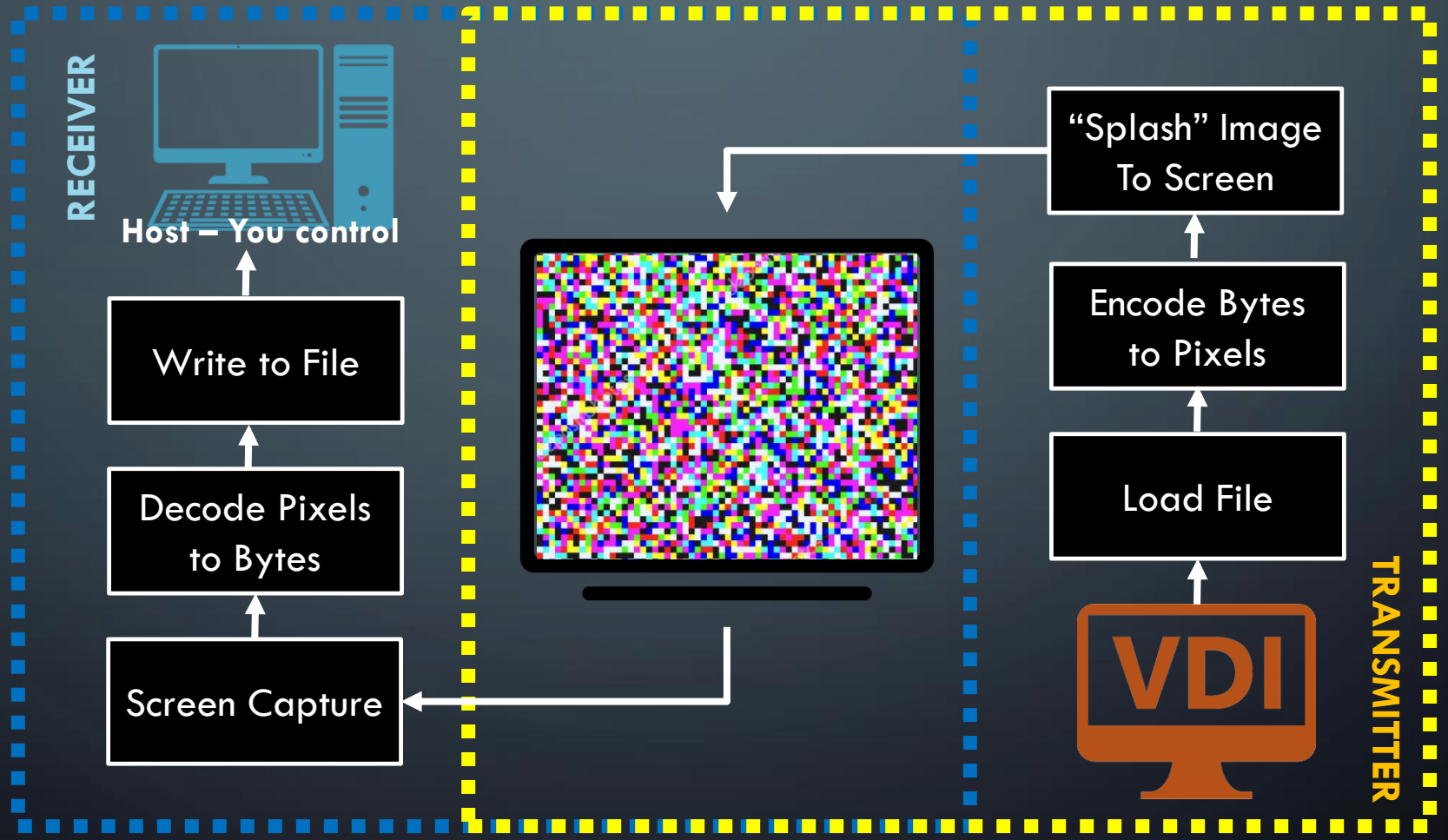
Host – You control



Decoded from image

VDI Session – You have access

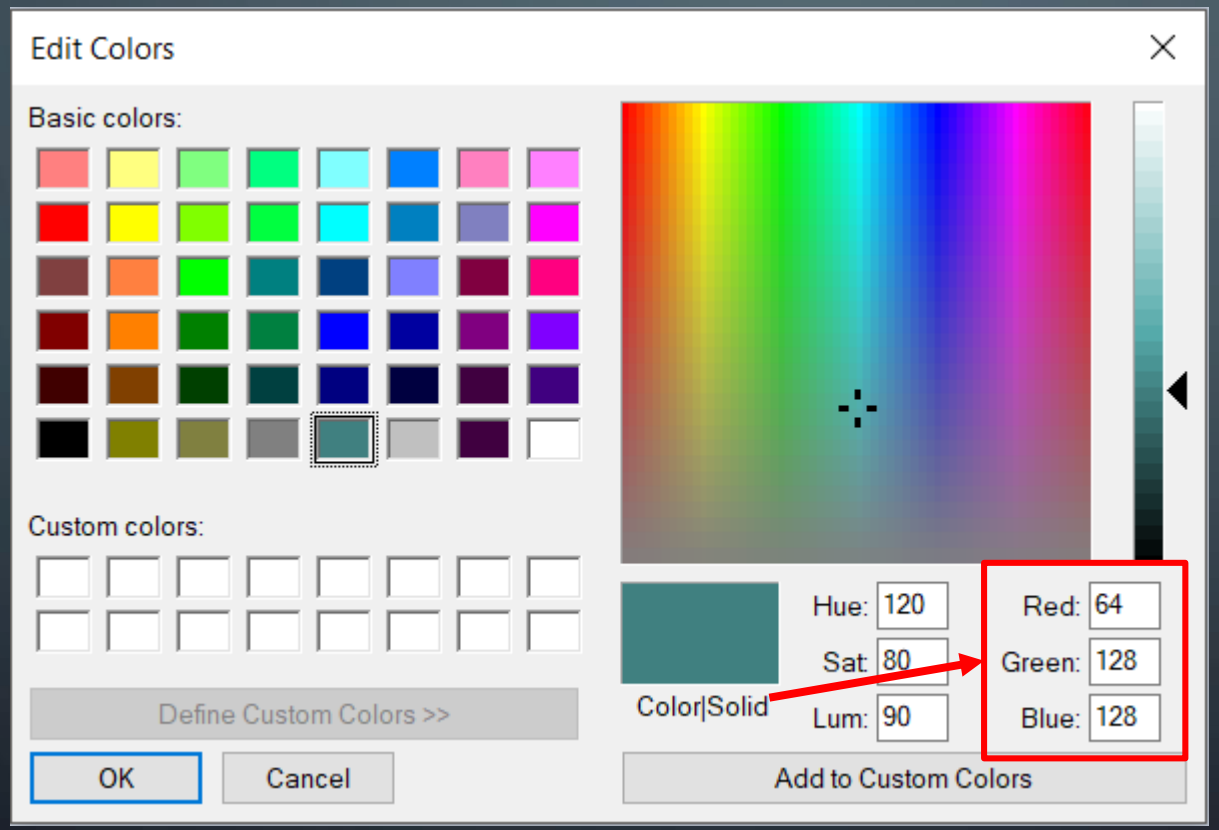
HIGH-LEVEL: STEPS



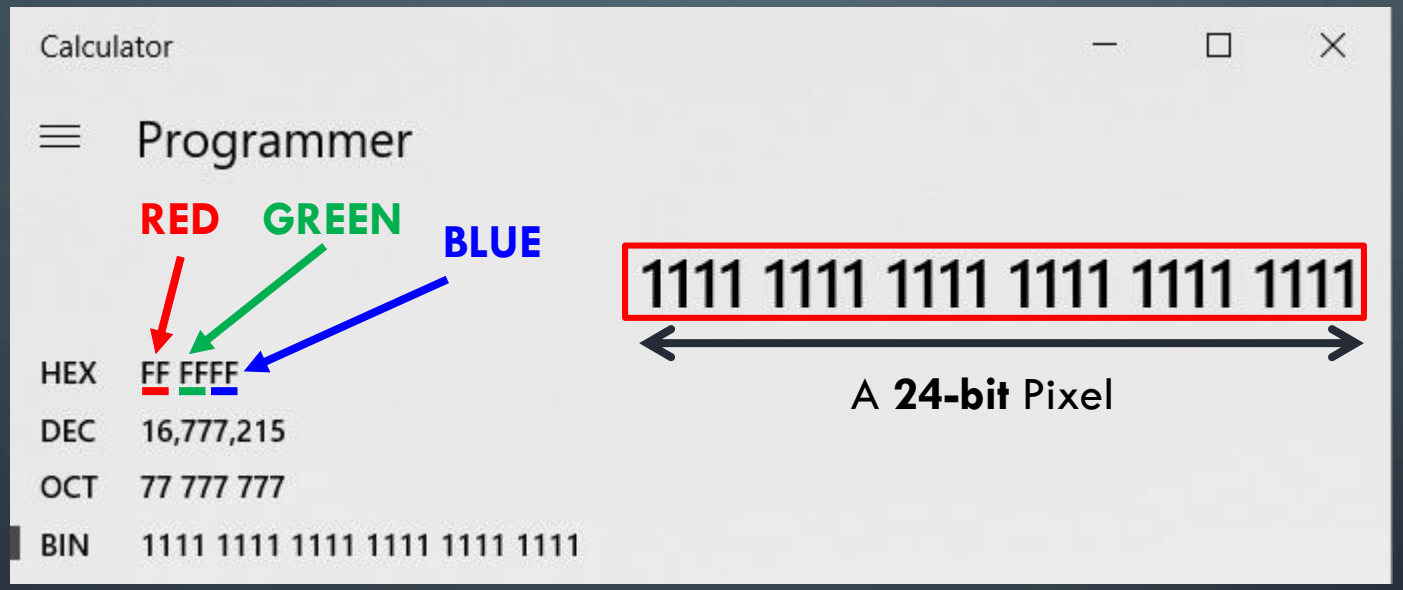
THE THEORY

- A computer screen is made of pixels
- For e.g. a standard resolution of 1080p, contains 1920 x 1080 = 2073600 pixels.
- Pixels are data. Each RGB pixel contains at least the Red, Blue & Green channel.
- In **24-bit color,** each channel's value from 0x00 to 0xFF
- In 32-bit color, RGBa is similar but with 'alpha' value or "transparency".

BASIC RGB


Edit Colors
Basic colors:
Custom colors:
Define Custom Colors >>
OK
Cancel
Color|Solid
Hue: 120
Sat: 80
Lum: 90
Red: 64
Green: 128
Blue: 128
Add to Custom Colors

BASIC COMPUTER SCIENCE

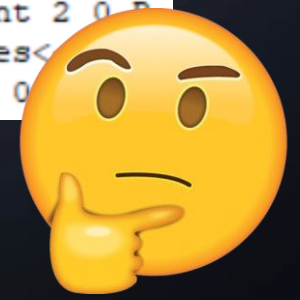


CONSIDER THIS…

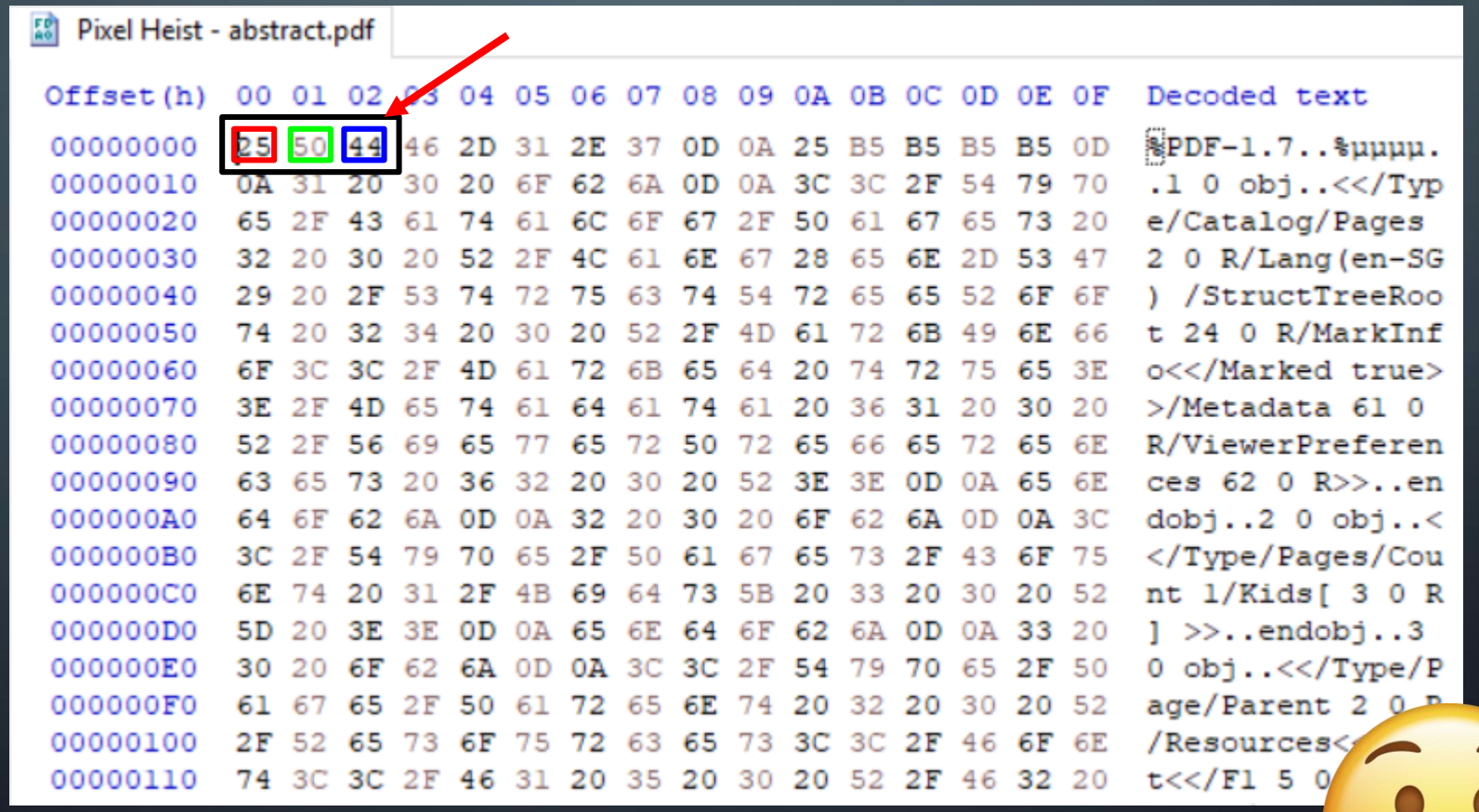
Pixel Heist - abstract.pdf

0x00 to 0xFF

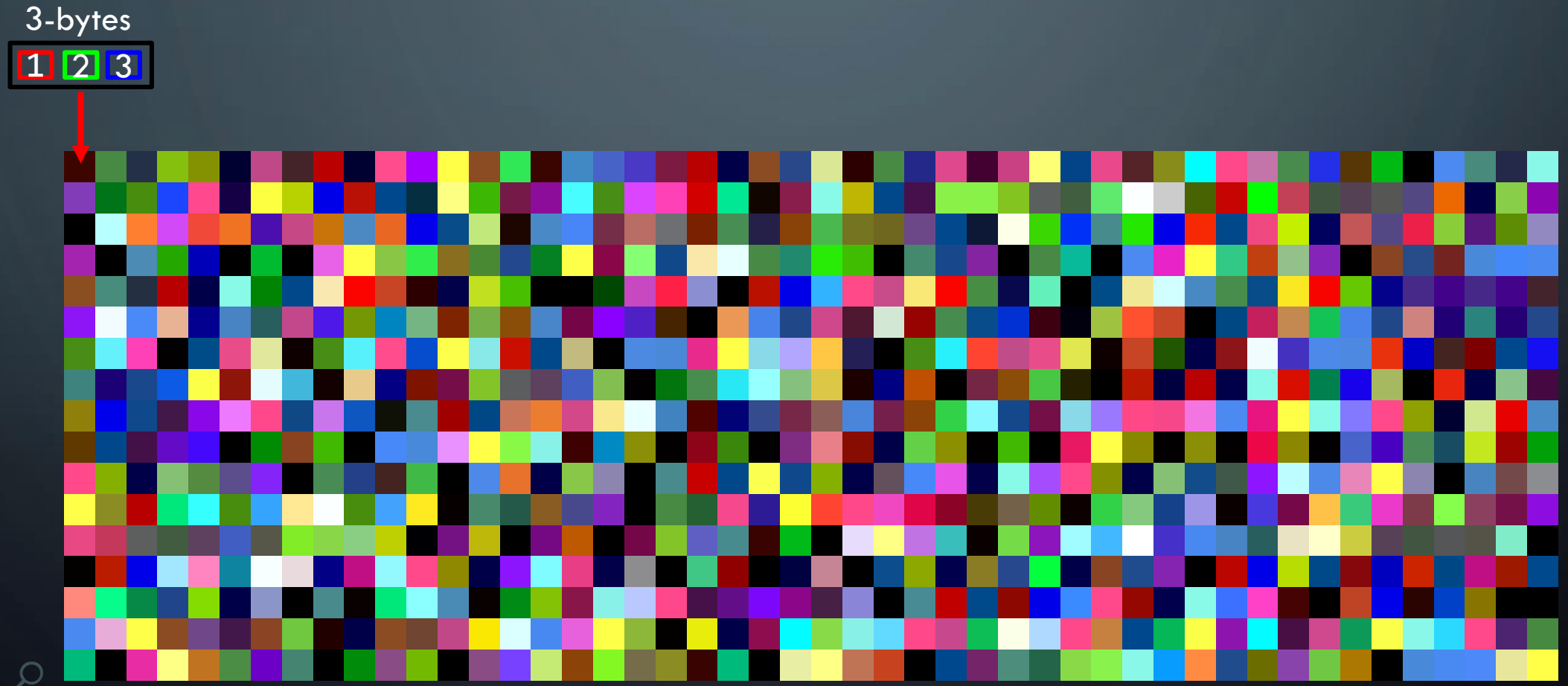
Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	25	50	44	46	2D	31	2E	37	0D	0A	25	B5	B5	B5	B5	0D	%PDF-1.7..%µµµµ.
00000010	0A	31	20	30	20	6F	62	6A	0D	0A	3C	3C	2F	54	79	70	.1 0 obj..<</Typ
00000020	65	2F	43	61	74	61	6C	6F	67	2F	50	61	67	65	73	20	e/Catalog/Pages
00000030	32	20	30	20	52	2F	4C	61	6E	67	28	65	6E	2D	53	47	2 0 R/Lang(en-SG
00000040	29	20	2F	53	74	72	75	63	74	54	72	65	65	52	6F	6F	) /StructTreeRoo
00000050	74	20	32	34	20	30	20	52	2F	4D	61	72	6B	49	6E	66	t 24 0 R/MarkInf
00000060	6F	3C	3C	2F	4D	61	72	6B	65	64	20	74	72	75	65	3E	o<</Marked true>
00000070	3E	2F	4D	65	74	61	64	61	74	61	20	36	31	20	30	20	>/Metadata 61 0
00000080	52	2F	56	69	65	77	65	72	50	72	65	66	65	72	65	6E	R/ViewerPreferen
00000090	63	65	73	20	36	32	20	30	20	52	3E	3E	0D	0A	65	6E	ces 62 0 R>>..en
000000A0	64	6F	62	6A	0D	0A	32	20	30	20	6F	62	6A	0D	0A	3C	dobj..2 0 obj..<
000000B0	3C	2F	54	79	70	65	2F	50	61	67	65	73	2F	43	6F	75	</Type/Pages/Cou
000000C0	6E	74	20	31	2F	4B	69	64	73	5B	20	33	20	30	20	52	nt 1/Kids[3 0 R
000000D0	5D	20	3E	3E	0D	0A	65	6E	64	6F	62	6A	0D	0A	33	20	] >>..endobj..3
000000E0	30	20	6F	62	6A	0D	0A	3C	3C	2F	54	79	70	65	2F	50	0 obj..<</Type/P
000000F0	61	67	65	2F	50	61	72	65	6E	74	20	32	20	30	20	52	age/Parent 2 0 R
00000100	2F	52	65	73	6F	75	72	63	65	73	3C	3C	2F	46	6F	6E	/Resources<<</Fon
00000110	74	3C	3C	2F	46	31	20	35	20	30	20	52	2F	46	32	20	t<</F1 5 0 R/F2



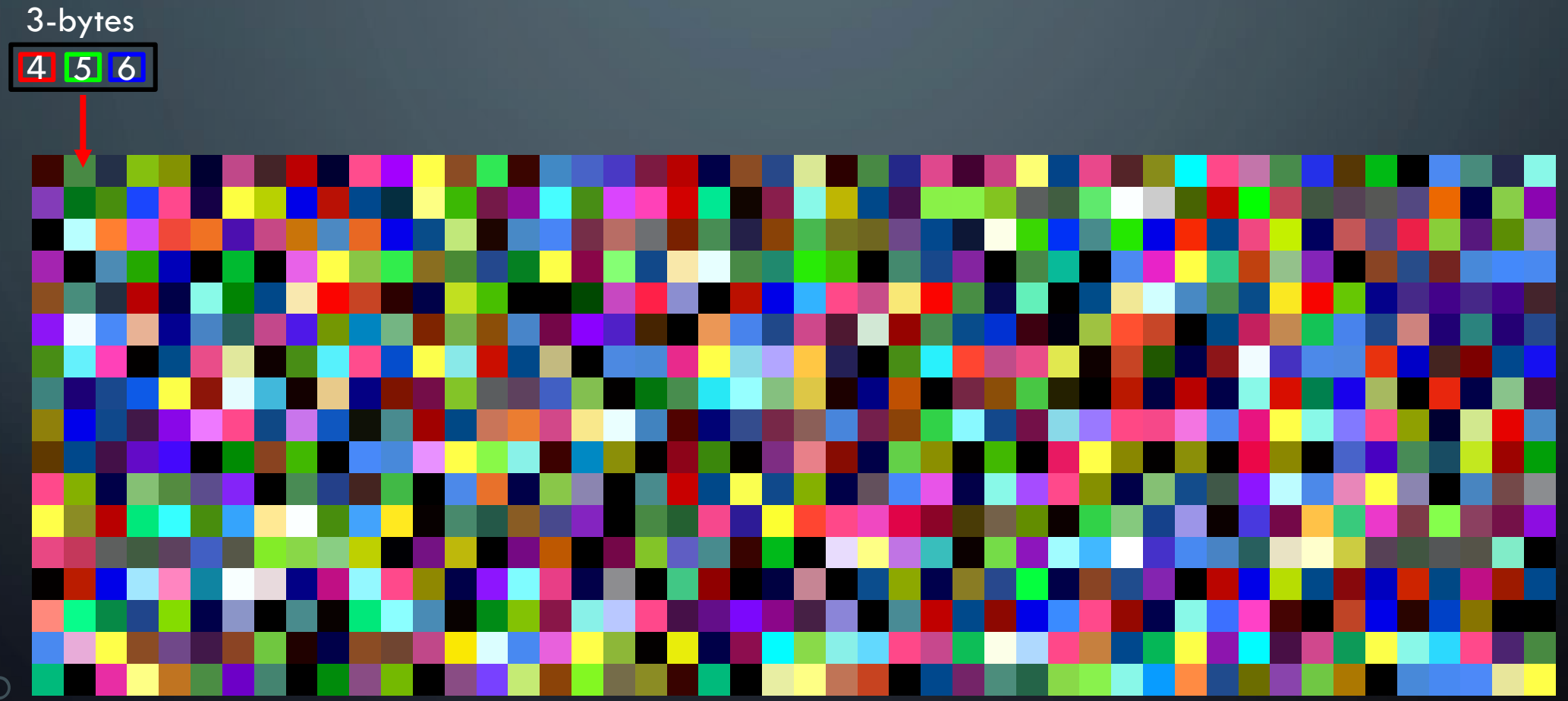
VOILÀ!


Pixel Heist - abstract.pdf
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
00000000 25 50 44 46 2D 31 2E 37 0D 0A 25 B5 B5 B5 B5 0D %PDF-1.7..%µµµµ.
00000010 0A 31 20 30 20 6F 62 6A 0D 0A 3C 3C 2F 54 79 70 .1 0 obj..<</Typ
00000020 65 2F 43 61 74 61 6C 6F 67 2F 50 61 67 65 73 20 e/Catalog/Pages
00000030 32 20 30 20 52 2F 4C 61 6E 67 28 65 6E 2D 53 47 2 0 R/Lang(en-SG
00000040 29 20 2F 53 74 72 75 63 74 54 72 65 65 52 6F 6F) /StructTreeRoo
00000050 74 20 32 34 20 30 20 52 2F 4D 61 72 6B 49 6E 66 t 24 0 R/MarkInf
00000060 6F 3C 3C 2F 4D 61 72 6B 65 64 20 74 72 75 65 3E o<</Marked true>
00000070 3E 2F 4D 65 74 61 64 61 74 61 20 36 31 20 30 20 >/Metadata 61 0
00000080 52 2F 56 69 65 77 65 72 50 72 65 66 65 72 65 6E R/ViewerPreferen
00000090 63 65 73 20 36 32 20 30 20 52 3E 3E 0D 0A 65 6E ces 62 0 R>>..en
000000A0 64 6F 62 6A 0D 0A 32 20 30 20 6F 62 6A 0D 0A 3C dobj..2 0 obj..<
000000B0 3C 2F 54 79 70 65 2F 50 61 67 65 73 2F 43 6F 75 </Type/Pages/Cou
000000C0 6E 74 20 31 2F 4B 69 64 73 5B 20 33 20 30 20 52 nt 1/Kids[3 0 R
000000D0 5D 20 3E 3E 0D 0A 65 6E 64 6F 62 6A 0D 0A 33 20] >>..endobj..3
000000E0 30 20 6F 62 6A 0D 0A 3C 3C 2F 54 79 70 65 2F 50 0 obj..<</Type/P
000000F0 61 67 65 2F 50 61 72 65 6E 74 20 32 20 30 20 52 age/Parent 2 0 R
00000100 2F 52 65 73 6F 75 72 63 65 73 3C 3C 2F 46 6F 6E /Resources<<
00000110 74 3C 3C 2F 46 31 20 35 20 30 20 52 2F 46 32 20 t<</F1 5 0

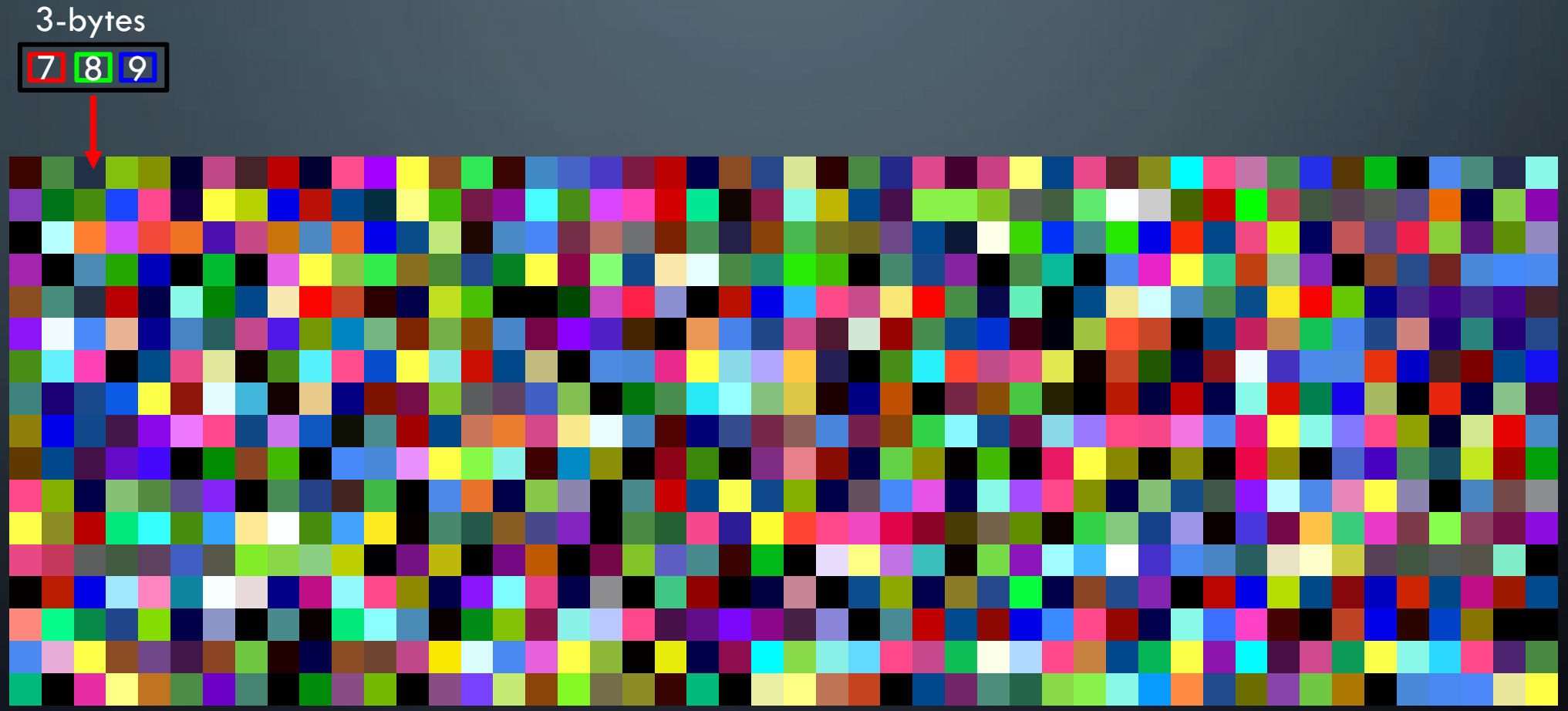
BYTES ENCODING TO PIXELS



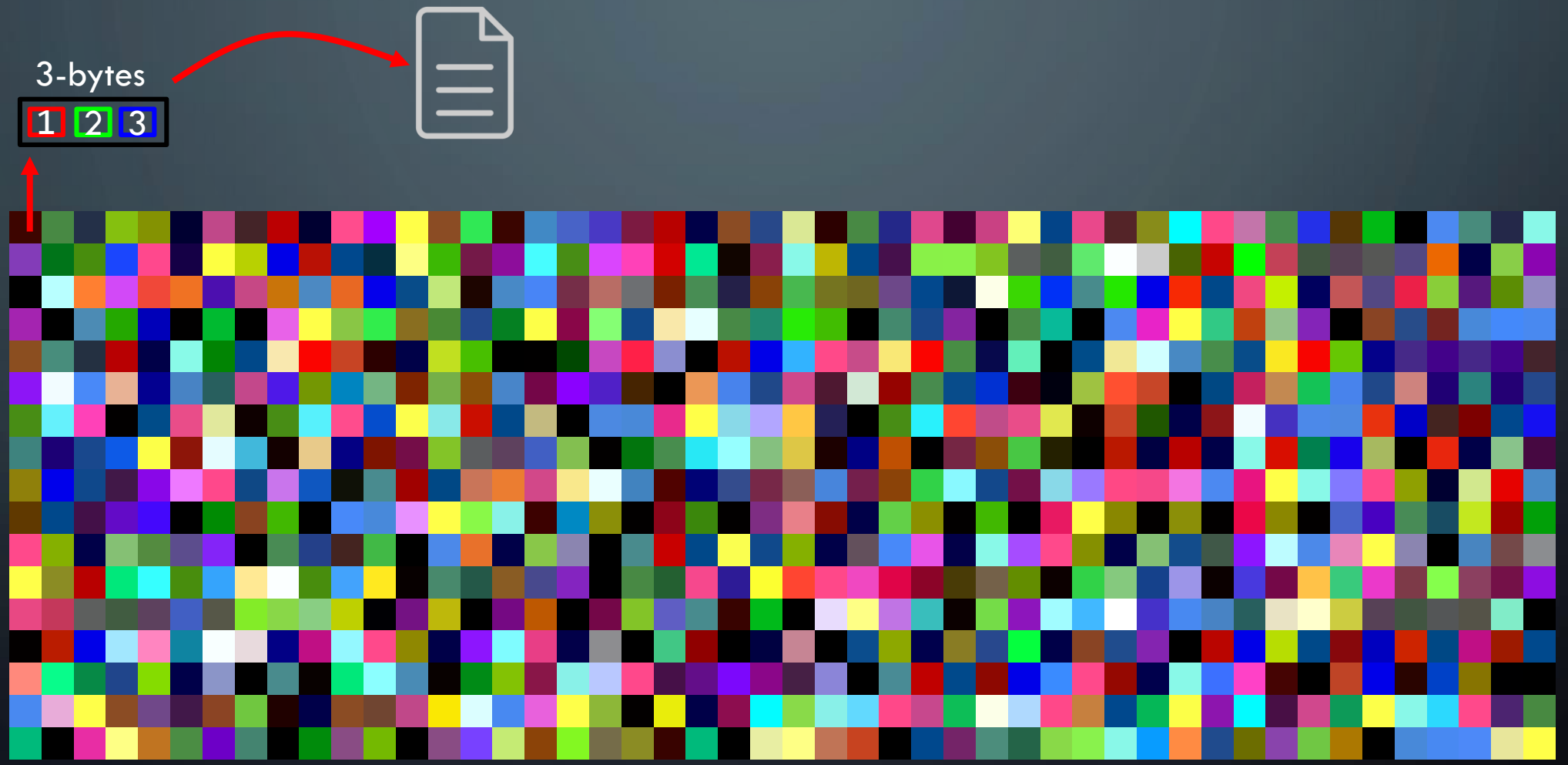
BYTES ENCODING TO PIXELS



BYTES ENCODING TO PIXELS



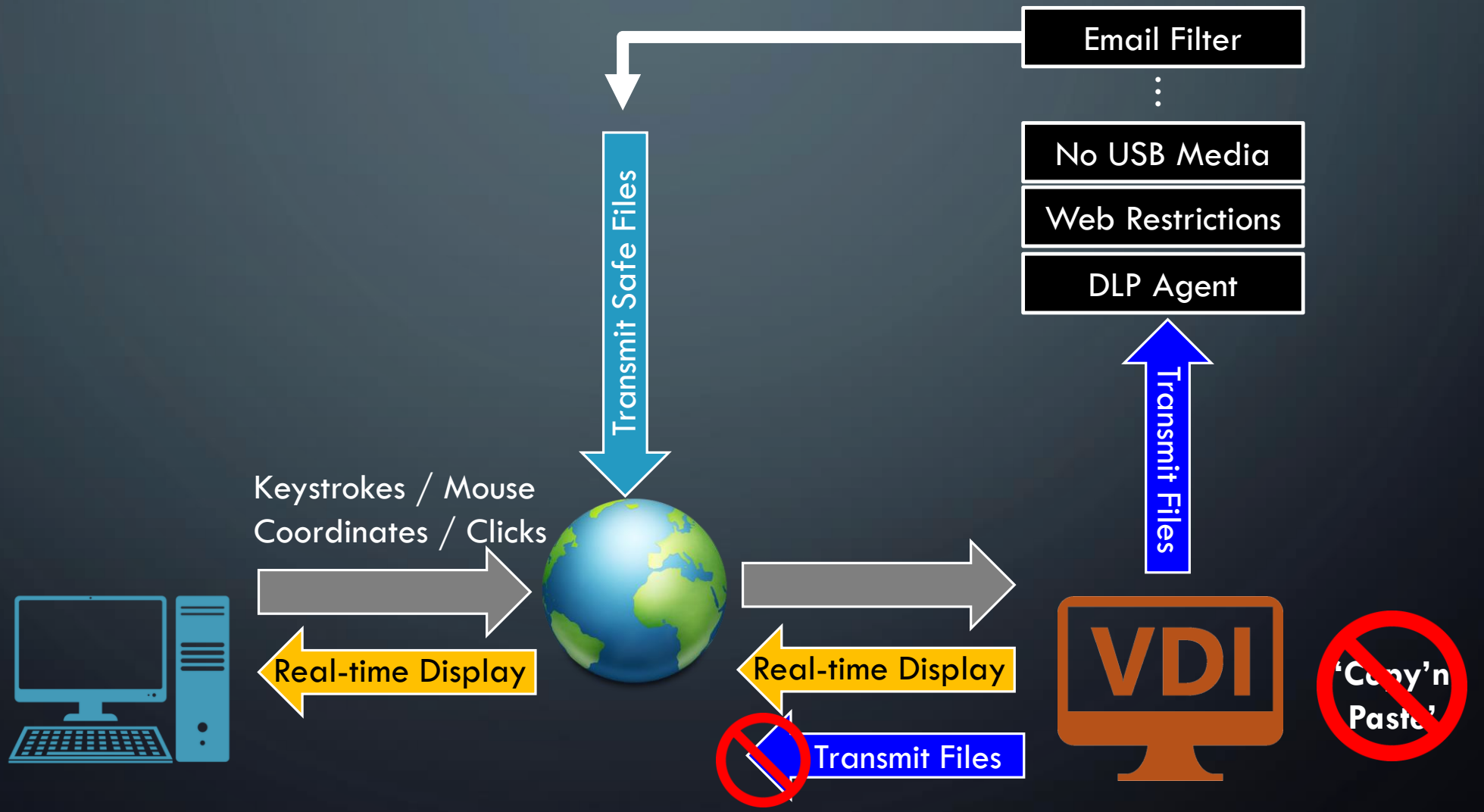
REVERSE! → DECODING PIXELS TO BYTES



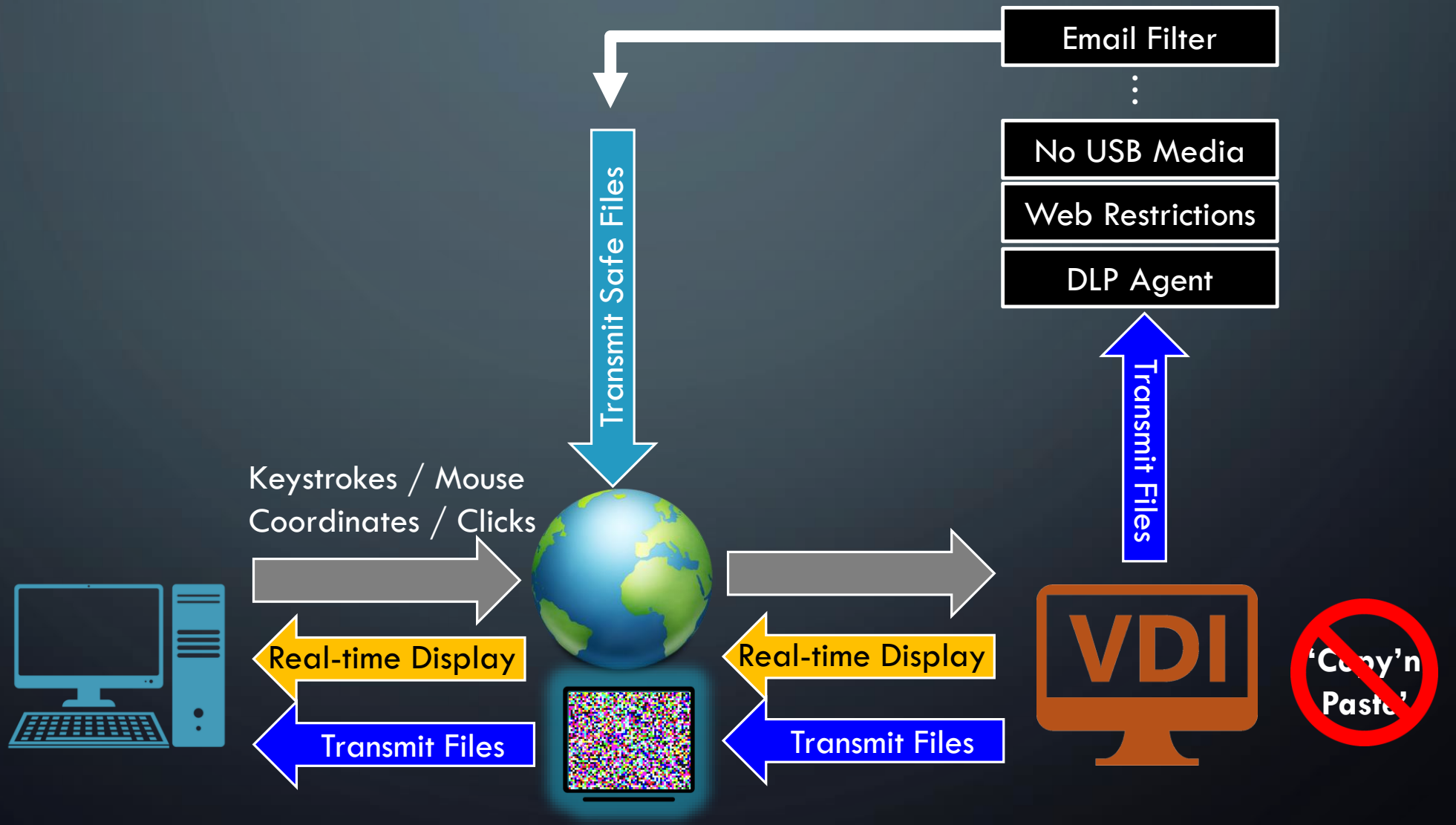
POTENTIAL IMPLICATIONS

- New exfiltration path for VDIs
- Breaks controls set in place for ‘Screen-only’ access. For example:
  - Administration via Jumpbox (e.g. RDP or VNC)
  - Access to Corporate environment over Internet via VDI implementations
- Difficult to prevent
- No known detective controls

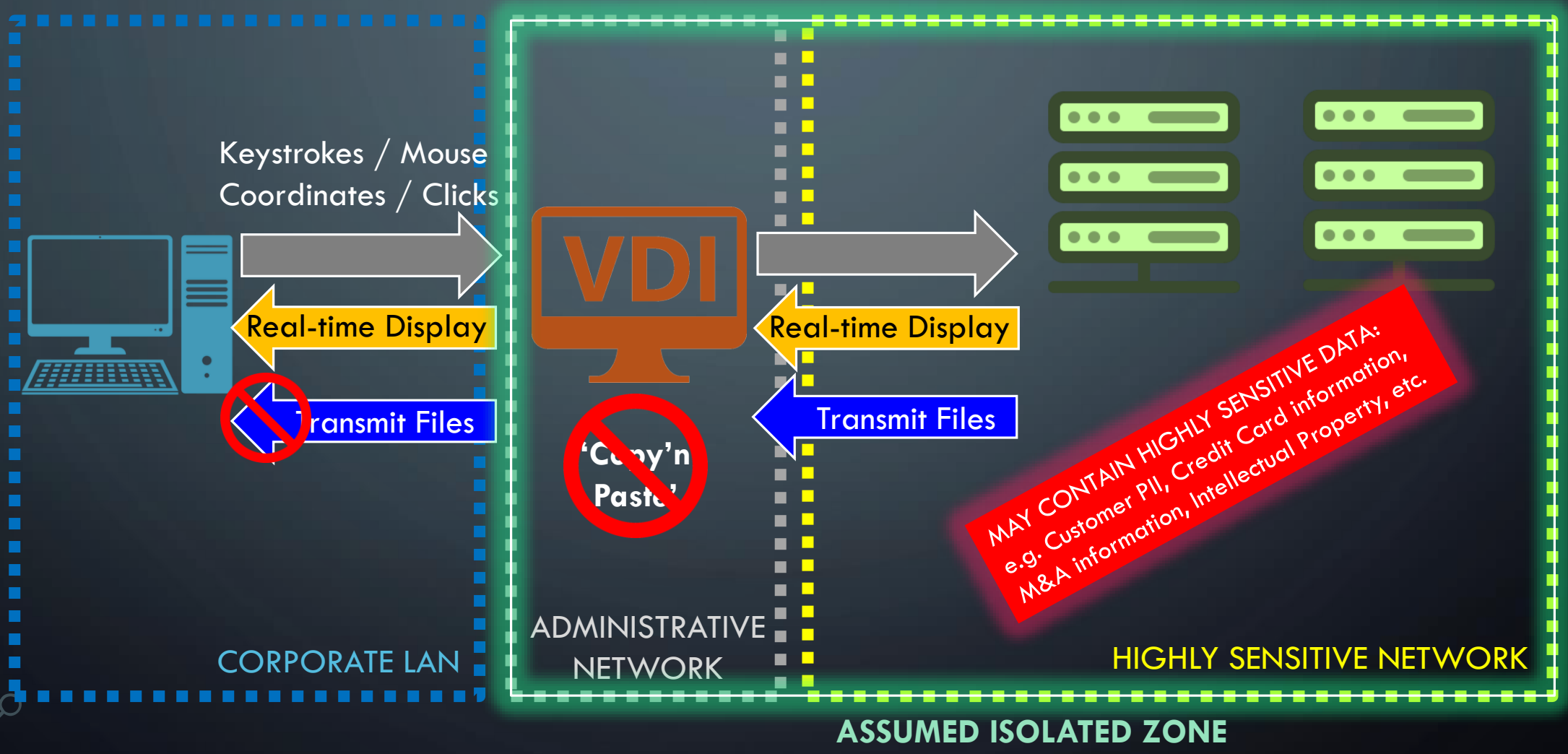
INTERNET VDI DLP ASSUMPTION



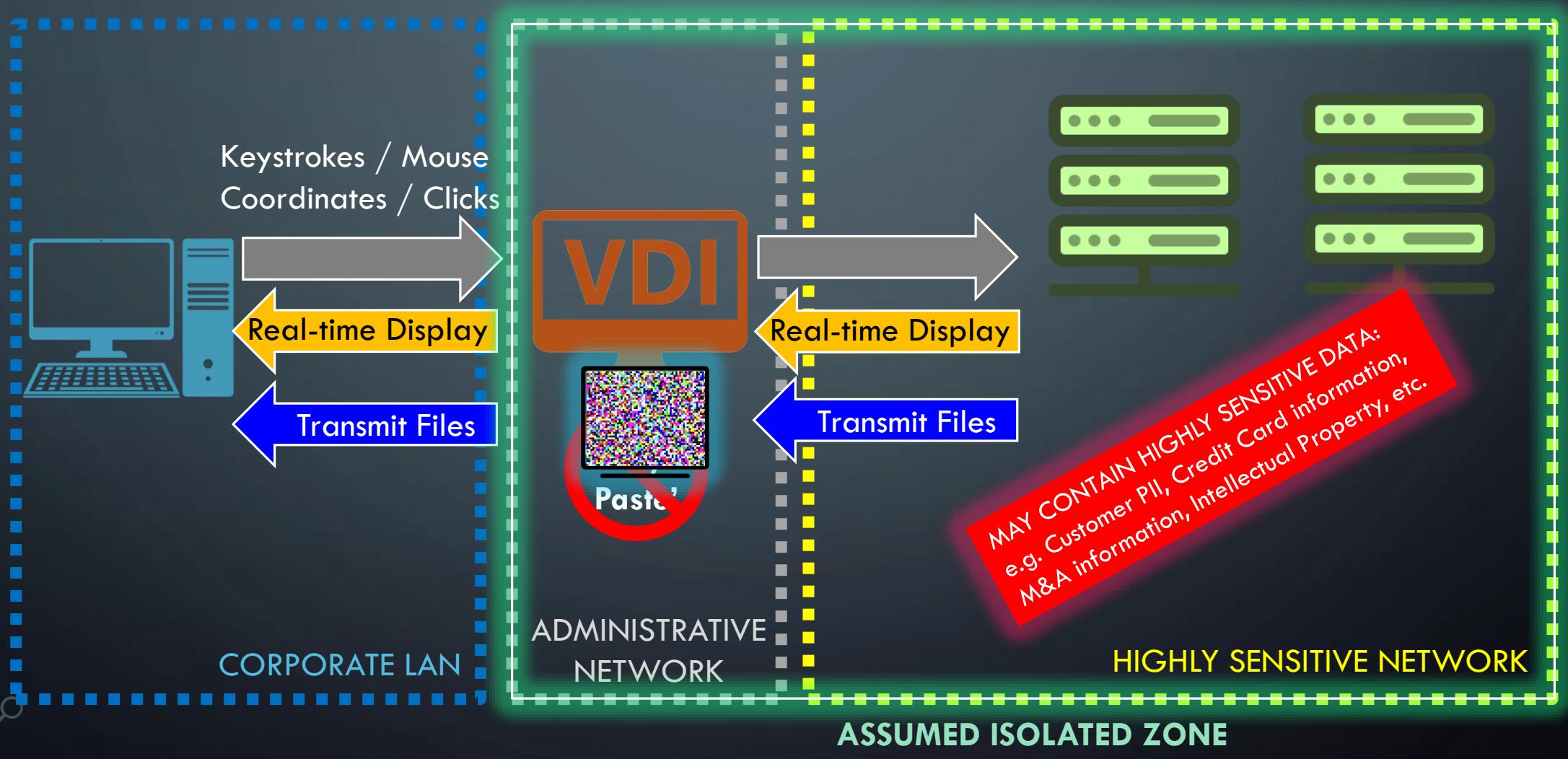
INTERNET VDI – PIXEL EXFIL.



ADMIN JUMPBOX ASSUMPTION



ADMIN JUMPBOX – PIXEL EXIL.



RUNNING PTP-RAT

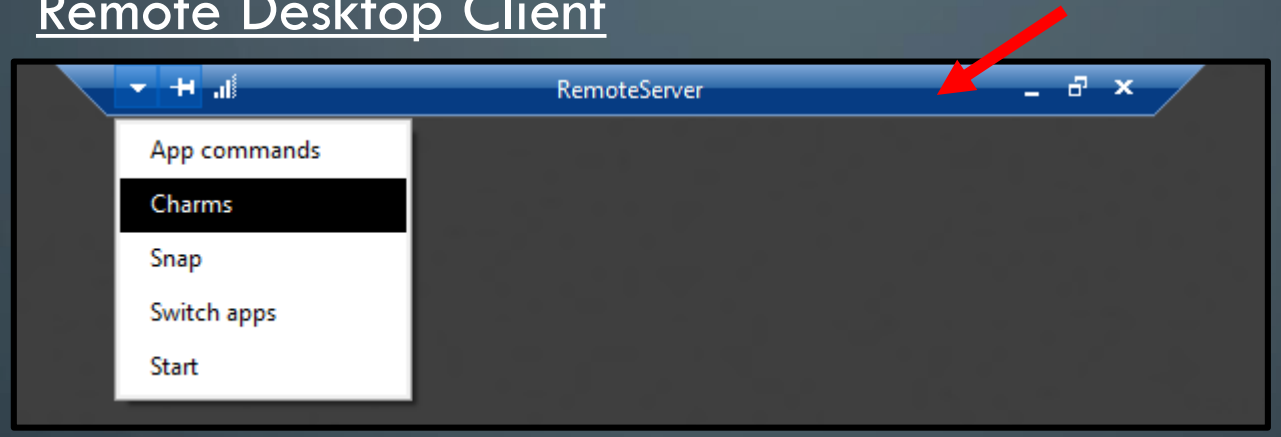
- Using the same file in two different locations
- LAN over RDP – **works**!
- Further testing over Internet on Citrix receiver – **failed**.
- Further testing over low-latency network over RDP – **failed**.
- What went wrong?

INITIAL ROOT CAUSE ANALYSIS

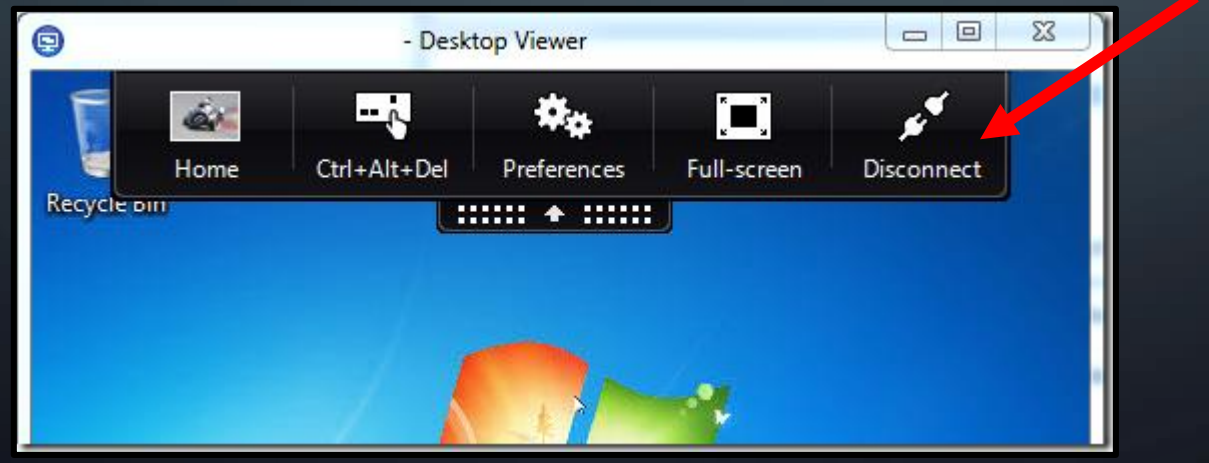
- PTP-RAT uses **FULL SCREEN** to transfer files
- Each pixel is 3-bytes of data. On-screen interferences corrupts data in transit.
- In-built client UI may be the culprits
- Another issue was discovered much later

ON-SCREEN "ARTIFACTS"

Remote Desktop Client



Citrix Receiver

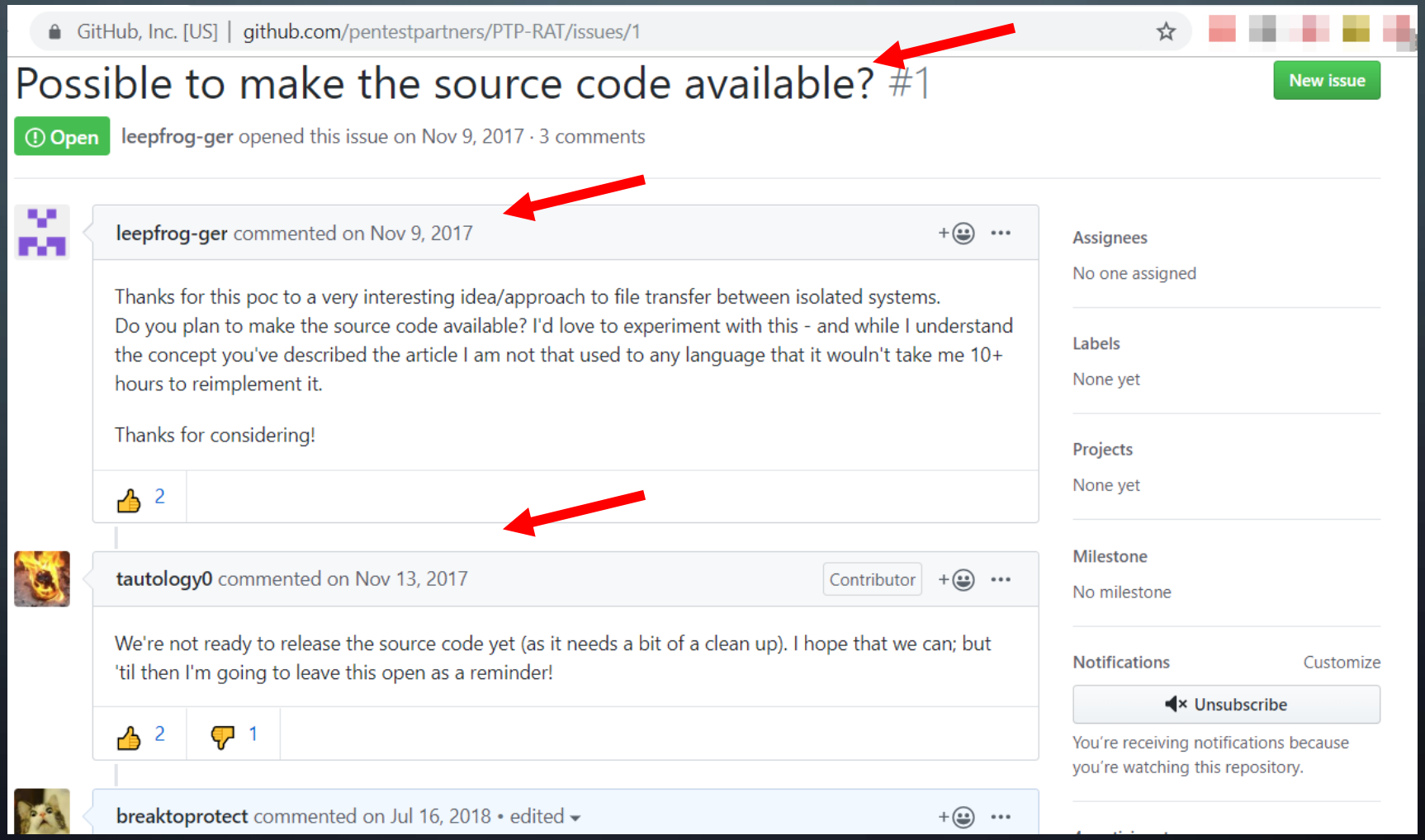


ON-SCREEN “ARTIFACTS”

SO… WE ARE SAFE THEN?

- Limitation of the tool != Limitation of the technology
- We cannot yet to conclude we are safe from such an exploit.
- So, we have to go deeper down the rabbit hole – either modify the tool's source code or make our own.

ATTEMPTS TO GET SOURCE CODE



'til then I'm going to leave this open as a reminder!

2 1

breaktoprotect commented on Jul 16, 2018 • edited

The tool is working flawlessly, except where in virtual environments the top notch (e.g. RDP's notch, or other virtualization notch) prevents the tool from working properly.

Is the source code ever going to be released? I'm contemplating to write a new one. But if the team intends to release the code as open source, I'm happy to request a pull later.

Don't let such an amazing project go to die ;)

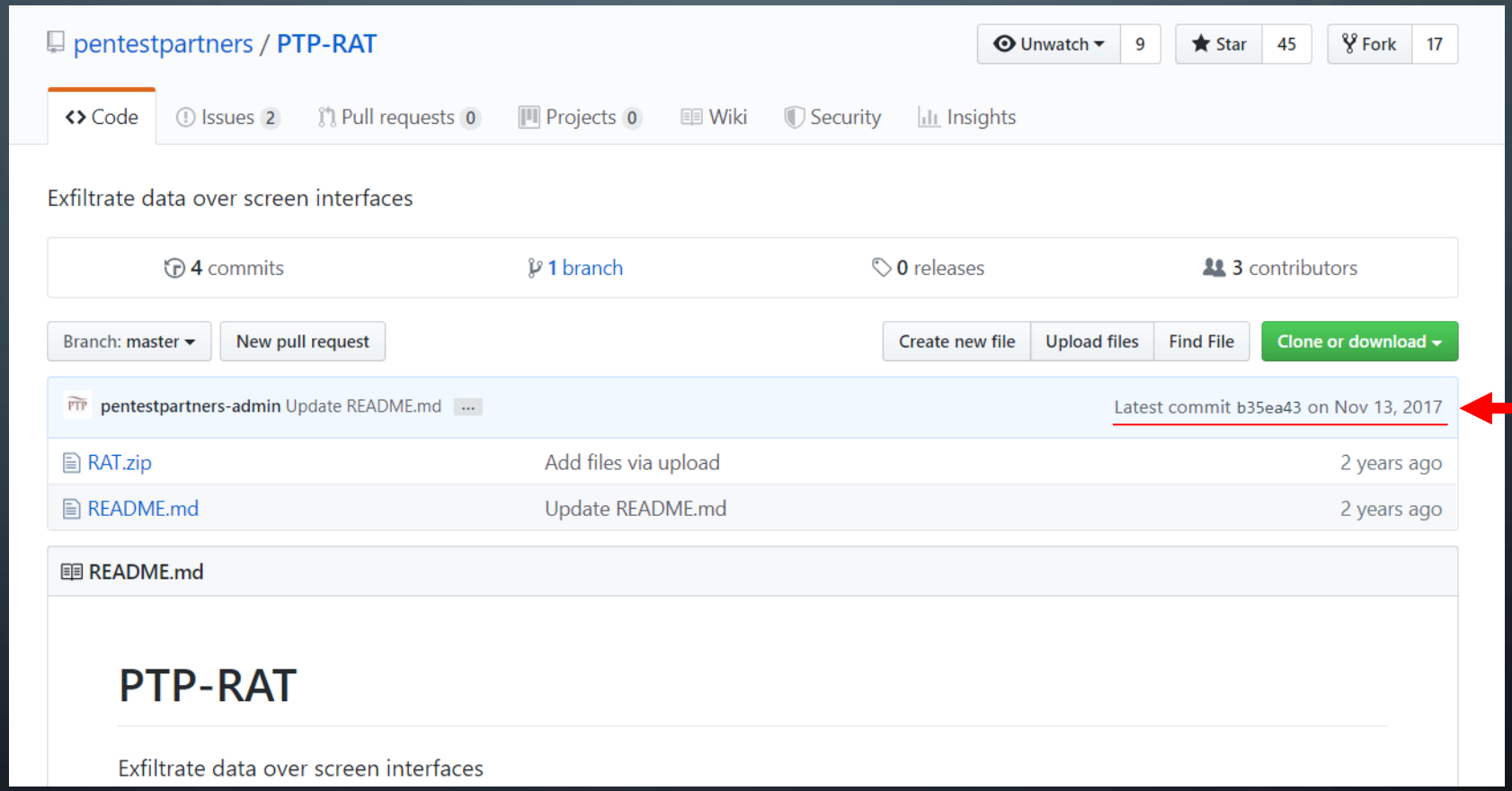
thEpisode commented on Jul 25, 2018

The idea to upload the source code is to improve as interested community :) Two heads (or more) are better than one

Write Preview

Leave a comment

NO LUCK ☹

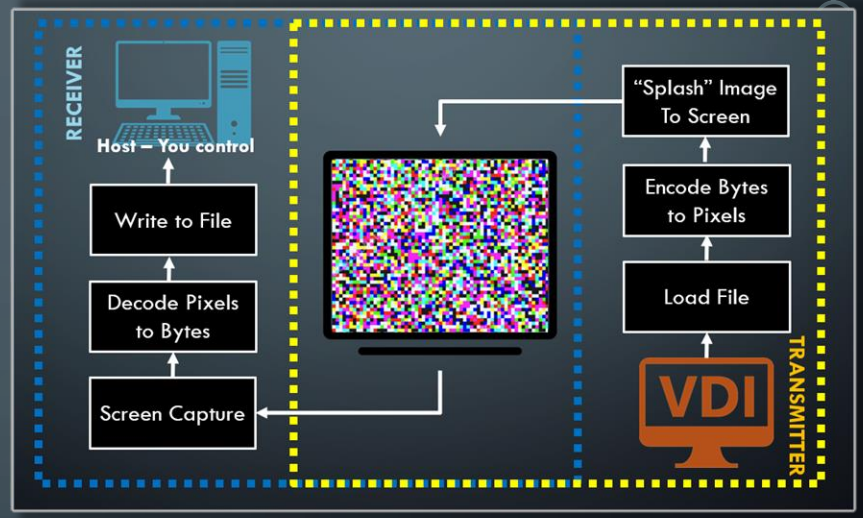


SCREENSHOT TAKEN IN SEP 2019

MAKING MY OWN PIXEL TOOL

- Follow all the steps (e.g. encode, splash, screenshot, decode)
- Implement it in a language (e.g. Python)
- Offset the Y axis down a little to compensate for UI artifacts
- Give it a name, say '**pixelcat**'. (inspired by 'netcat')
- Then test it again!

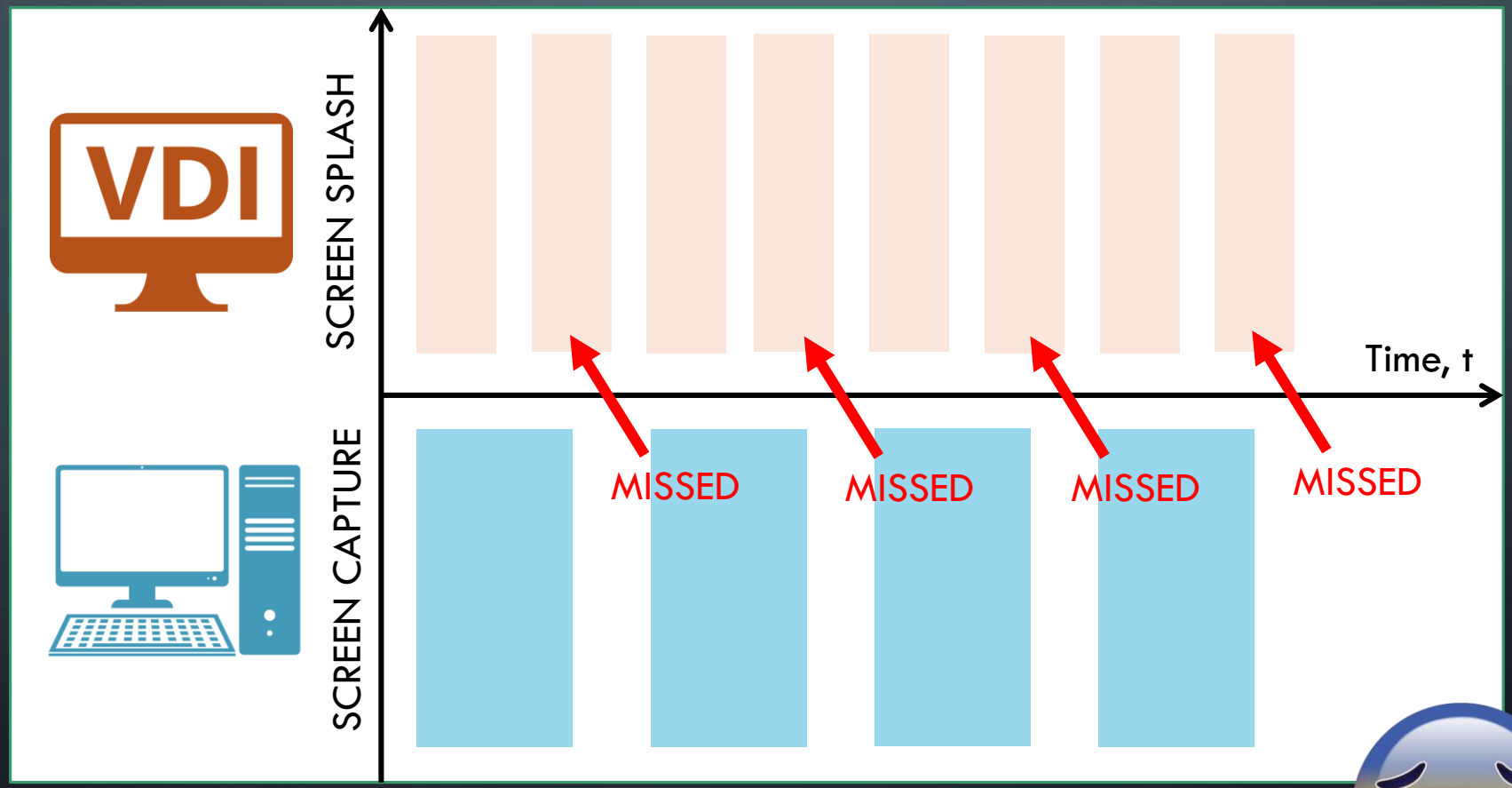
BUILDING A "TRANSPORT PROTOCOL"



Areas to consider:

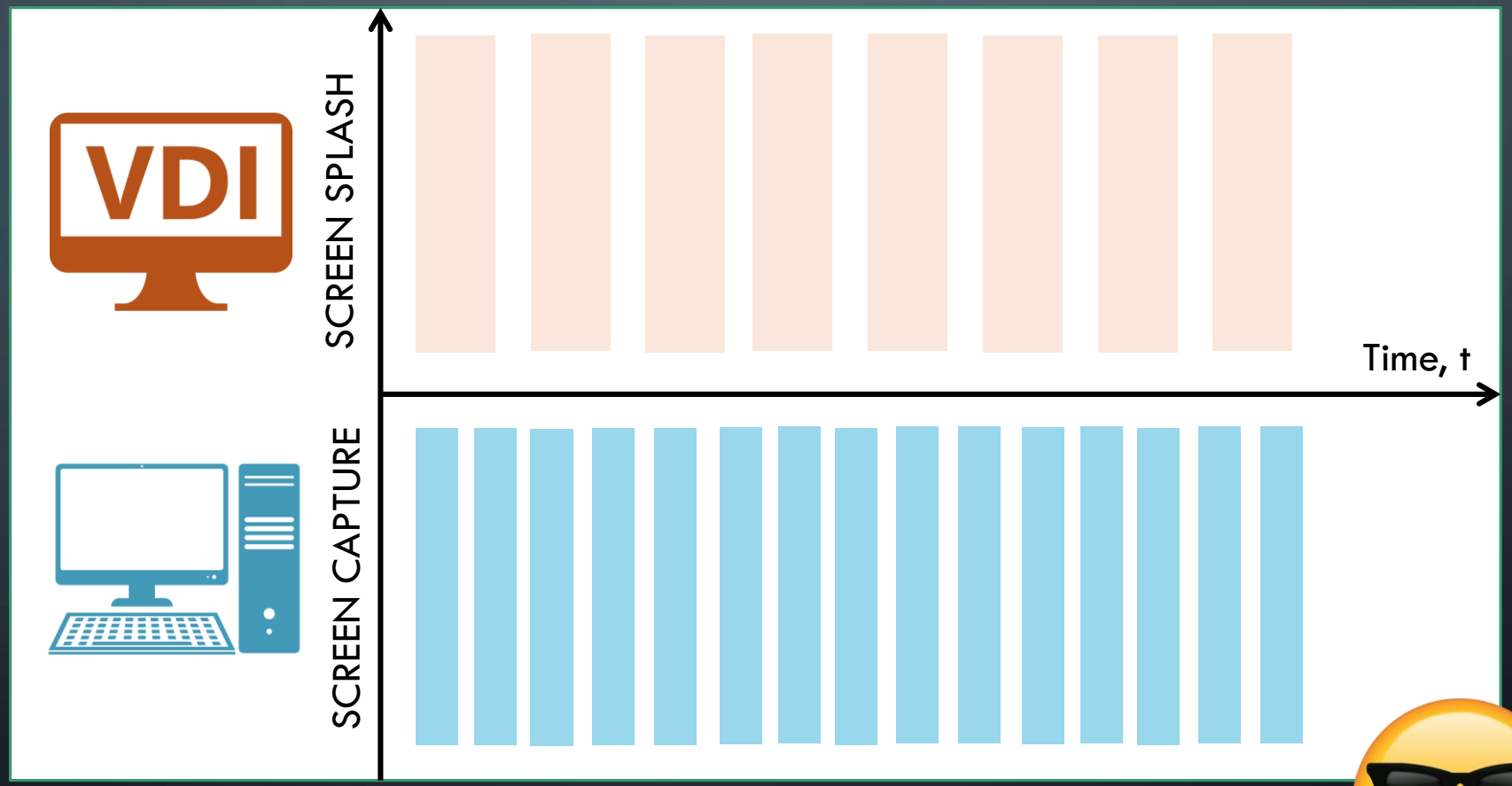
- "Splash" and screen capture frequency/rates.
  - Note: The client and server are not synchronized
- Signature to uniquely identify pixelcat's "splash" images
- Page control: A file may take more than 1 screen splash
- Meta information: Filename, extensions, etc.
- On-screen artifacts: The 'notch' at the top, mouse pointer, etc.

SCREEN SPLASH & CAPTURE RATE



A CASE OF UNDER-SAMPLING

SCREEN SPLASH & CAPTURE RATE



OVER-SAMPLING!

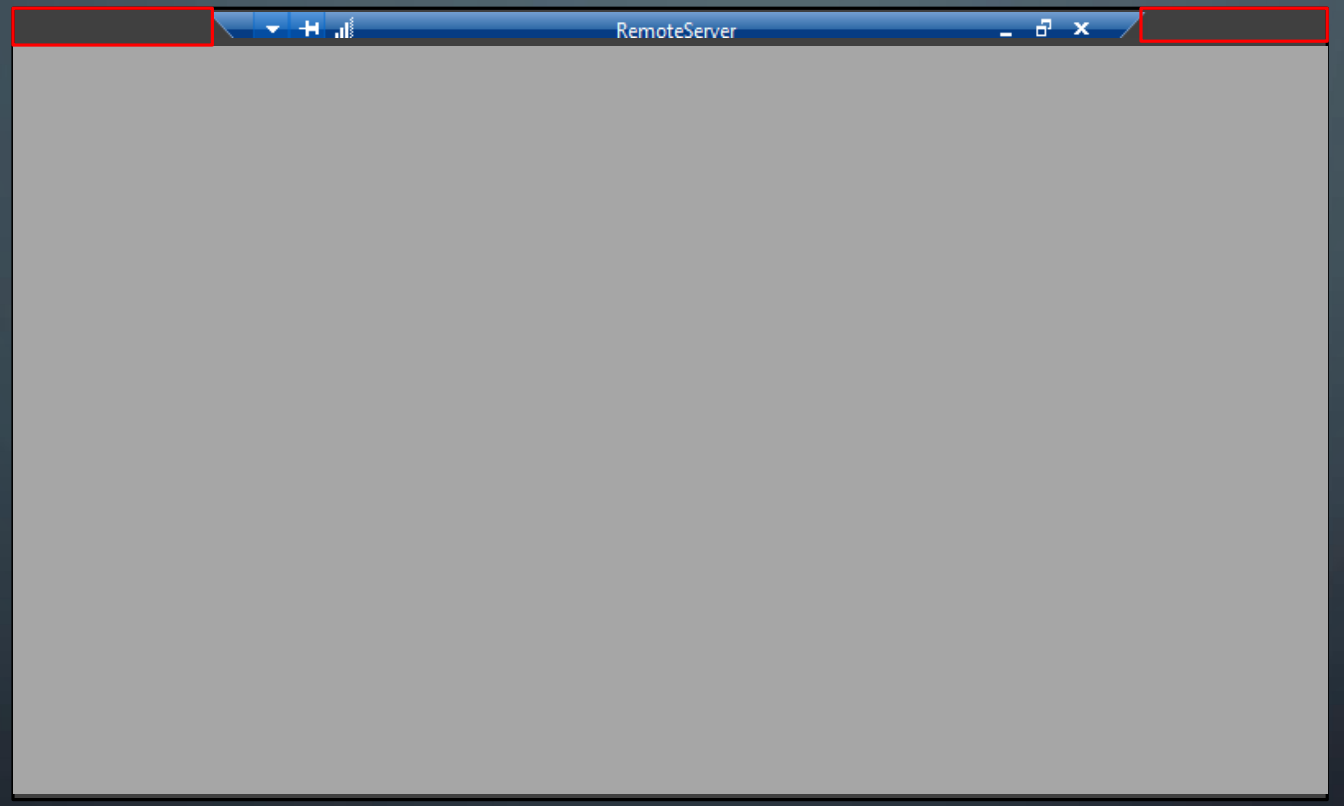
WHAT ABOUT DUPLICATES? SEQUENCES? LOST FRAMES?

- Keep track of splash images received.
  - Remove duplicates
- Keep track of sequence of splash images received.
  - E.g. Page 0, 1 2, 3 4
- Careful with CPU time. Do post-delivery processing – avoid on-the-fly processing.
  - Post-processing: Focus on capturing images. Decode later.
  - On-the-fly processing: For each image captured, decode immediately.
  - Or do async / parallel processing.

HEADERS

- Signature – To allow pixelcat to identify screen splashes
- Current Page – Allow pixelcat to remove duplicates and sequence the pages in order
- Max/Last Page – Allow pixelcat to know when to terminate screen capture
- Filename with extension – To allow pixelcat to reconstruct the file with original filename

HEADERS



HEADERS

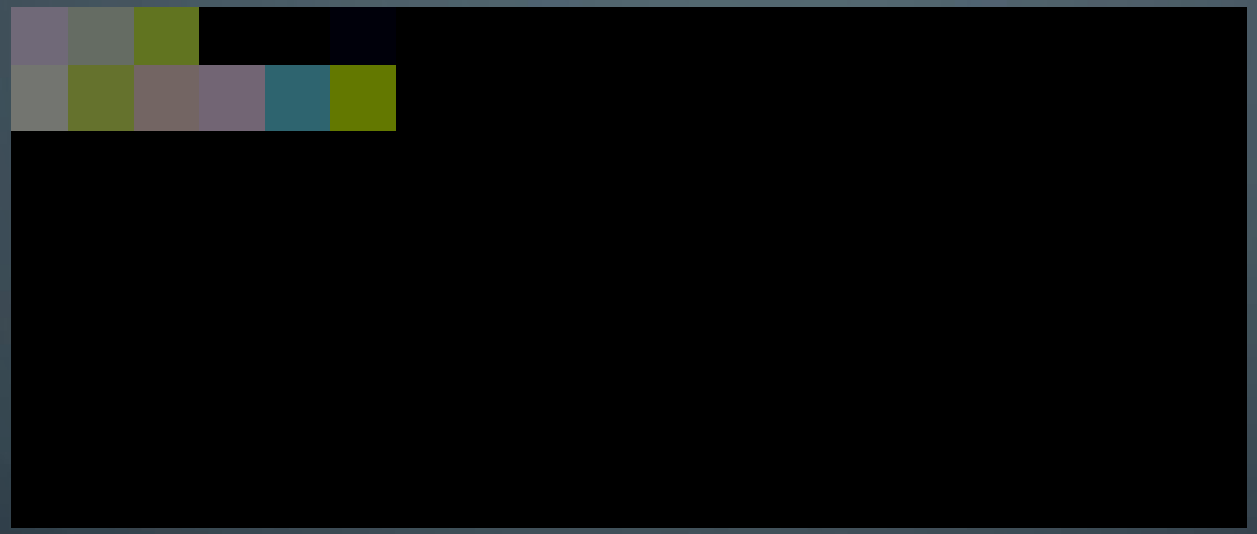
[70 69 78]	[65 6c 63]	[61 74 20]		[0,0,0]	[0,0,A]		
[73 75 70]	[65 72 2d]	[73 65 63]	[72 65 74]	[2e 64 6f]	[63 78 00]		

Signature

Current Pg Last Page

p i x	e l c	a t		(Page) 0	(of) 10		
s u p	e r -	s e c	r e t	.do	c x (NULL)		
		Filename & Extension					

HEADERS – LOOKS LIKE…



p i x	e l c	a t		(Page) 0	(of) 10		
s u p	e r -	s e c	r e t	.do	c x (NULL)		

HOW FAST CAN IT GO?

- Depends largely on bandwidth
- Bandwidth depends on the network, your access bandwidth, and the VDI
- Each splash screen data size on a Standard monitor 1920 x 1080:
  - 1920 x 1080 x 3 = 6220800 bytes or **5 MBs**
- Attempting to transfer 5MB images per second via VDI with screen display rate at 500kbps – won't work ☺
- Larger screens (e.g. 3440x1440 ultrawide) will suck up more bandwidth
- "Why don't we just whip out the phone's camera and take picture?"

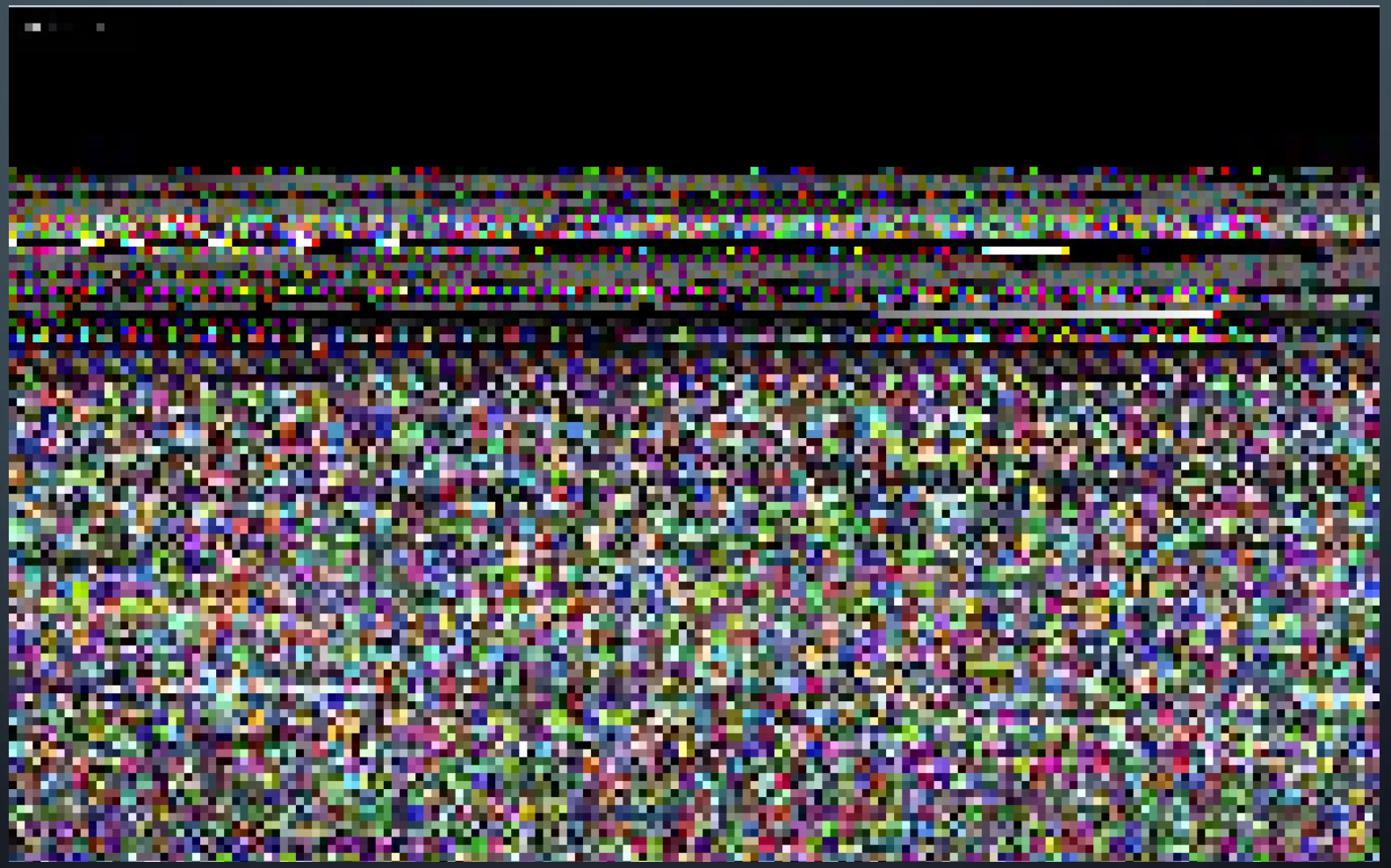
FURTHER TEST ANALYSIS

- At low-bandwidth, VDI technology adopts compression to reduce latency
- Graphics compression does not affect the visual much
- However, it's catastrophic for data encoded in pixel. Any slight change would corrupt the file.
- Let's examine with a A-B comparison.

COMPRESSION ARTIFACTS – A: NO COMPRESSION

NO COMPRESSION – ACTUAL FILE

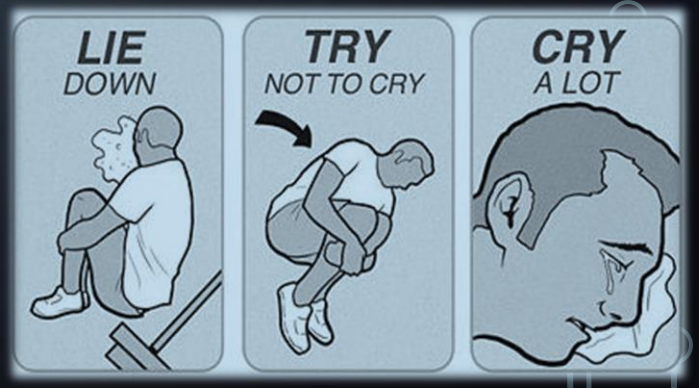
COMPRESSION ARTIFACTS – B: WITH COMPRESSION



SAME FILE – VIEWED FROM VDI SCREEN

LESSONS FROM PIXEL EXFIL. EXPERIMENT

- File transfer over LAN RDP is **reliable**!
- Overcoming notches… **possible**!
- File transfer with pixel exfiltration technique over low-bandwidth network introduces compression artifacts
- **Compression artifacts absolutely destroys data integrity…GGWP?**



CAN WE MAKE IT MORE RELIABLE?

- The standard byte-per-RGB-channel encoding technique requires **lossless network transmission.**
- Beg administrators to disable/turn off compression?
- Expecting lossless transmission over Internet serving large enterprises (10K – 200K employees) – not going to happen soon.
- But we give it another shot.

ANTI-COMPRESSION? TECHNIQUE

- Instead of byte-per-RGB-channel, bit per pixel.
- Capacity to transfer reduced significantly:
  - Byte-per-RGB-channel technique -> ~5 MB per page
  - Bit per pixel technique -> ~253 KB per page
- Concept remains the same – only encoding style changed

ANTI-COMPRESSION? TECHNIQUE
BIT PER PIXEL

Pixel Heist - abstract.pdf

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	25	50	44	46	2D	31	2E	37	0D	0A	25	B5	B5	B5	B5	0D	%PDF-1.7..%µµµµ.
00000010	0A	31	20	30	20	6F	62	6A	0D	0A	3C	3C	2F	54	79	70	.1 0 obj..<</Typ

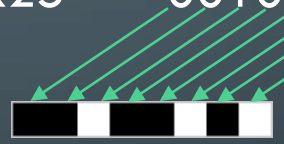
Byte Per RGB Channel

0x25

(25, …, …)

Bit Per Pixel

0x25 == 00100101



(0,0,0) (0,0,0) (255,255,255) (0,0,0) …

ANTI-COMPRESSION? TECHNIQUE #1
BIT PER PIXEL – BINARY LOGIC

Bit Per Pixel

0x25 == 00100101

(0,0,0) (0,0,0) (255,255,255) (0,0,0) …

- Allows a large margin of error to overcome potential compression artifacts.
- Two-levels of comparison:
  - Channel-level (0-127) = Black; (128-255) = White
  - Pixel-level:
    - if (B, B, W) means bit 0.
    - If (W, B, W) means bit 1.

COMPARISON ARTIFACTS AC MODE

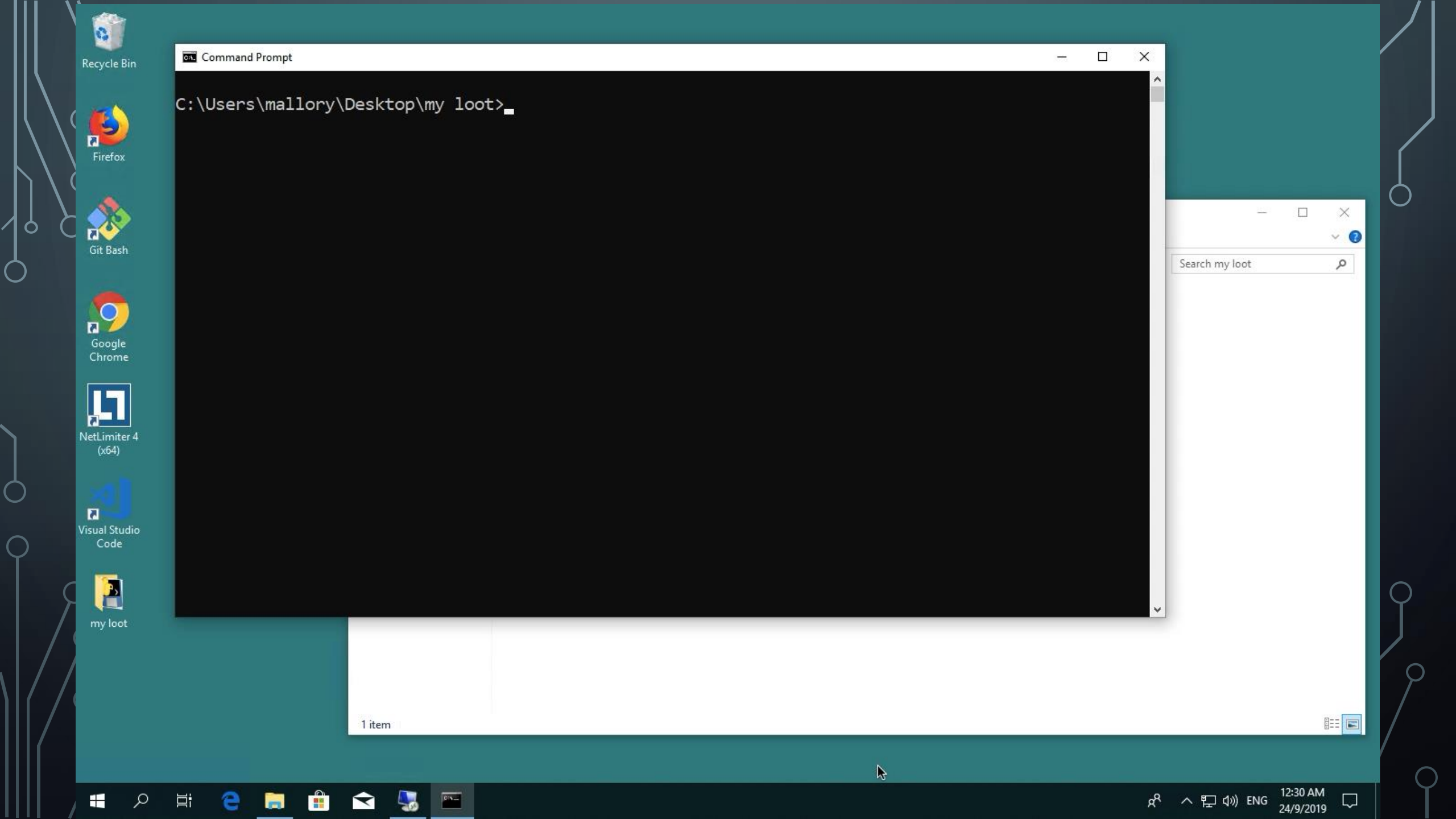
NO COMPRESSION – ACTUAL FILE

COMPARISON ARTIFACTS AC MODE

SAME FILE – VIEWED FROM VDI SCREEN

DEMO - RDP

- Green desktop – Receiver
- Blue desktop – Sender / Transmitter
- 1MB file
- Anti-compression mode

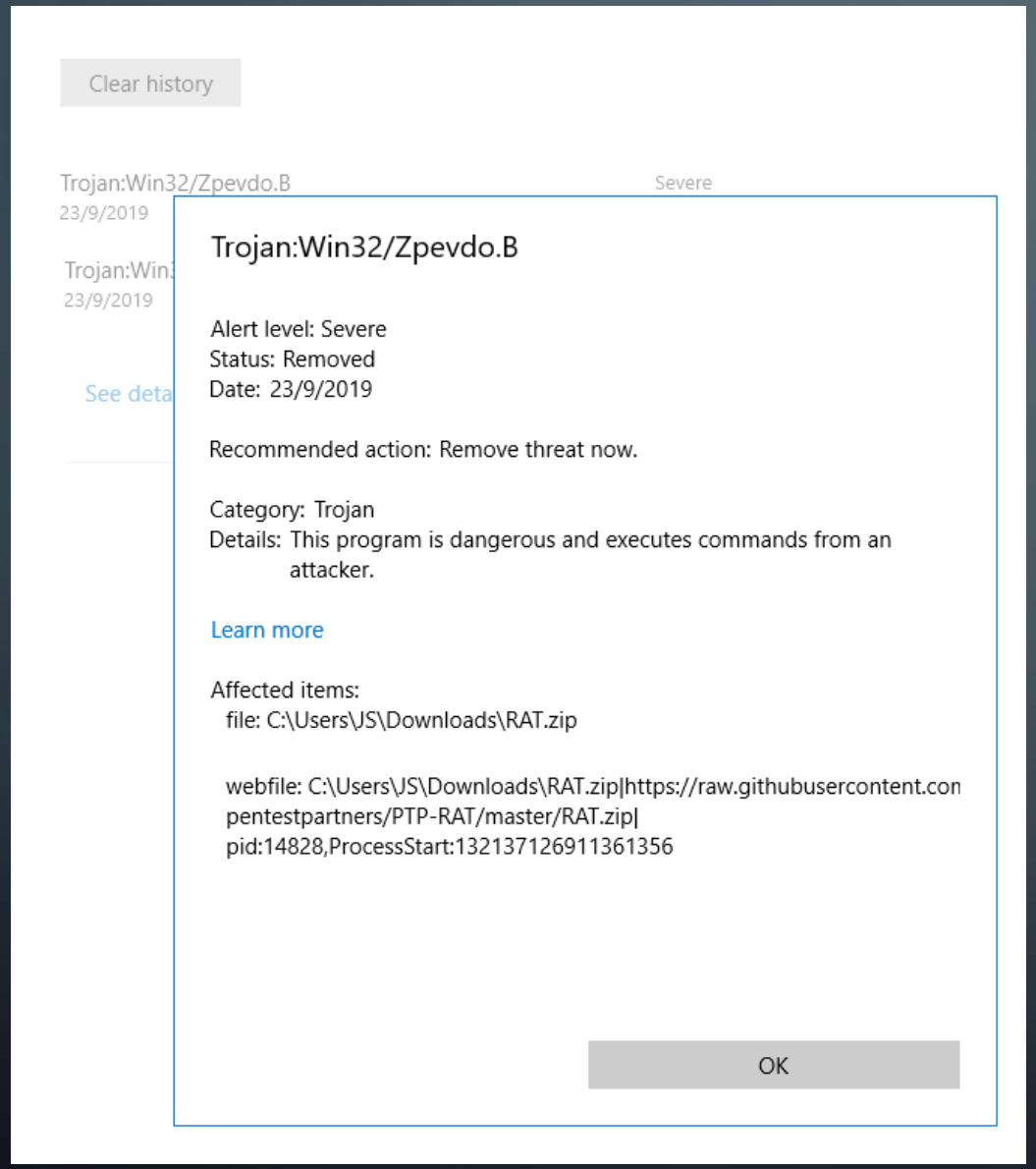

Recycle Bin
Firefox
Git Bash
Google Chrome
NetLimiter 4 (x64)
Visual Studio Code
my loot
Command Prompt
C:\Users\mallory\Desktop\my loot>
Search my loot
1 item
ENG
12:30 AM
24/9/2019

DOES IT WORK?

- Yes…ish **~98%** accuracy.
- Slower as compared to full RGB encoding 250KB vs 5MB per page.
- Useful in transferring error-tolerant files (e.g. text)
  - Source Code, Credit Card numbers, PII, etc.
- Unable to transfer binaries:
  - Minor alteration in program == FATAL ERROR
- PDF, DOCX, XLSX, etc. ← If you're lucky…

PREVENTION?

- Application Control – prevent unknown binaries from executing.
  - Difficult to implement.
  - Not realistic. (e.g. running scripts like python, powershell, Vbscripts, etc may circumvent this)
- Anti malware…?

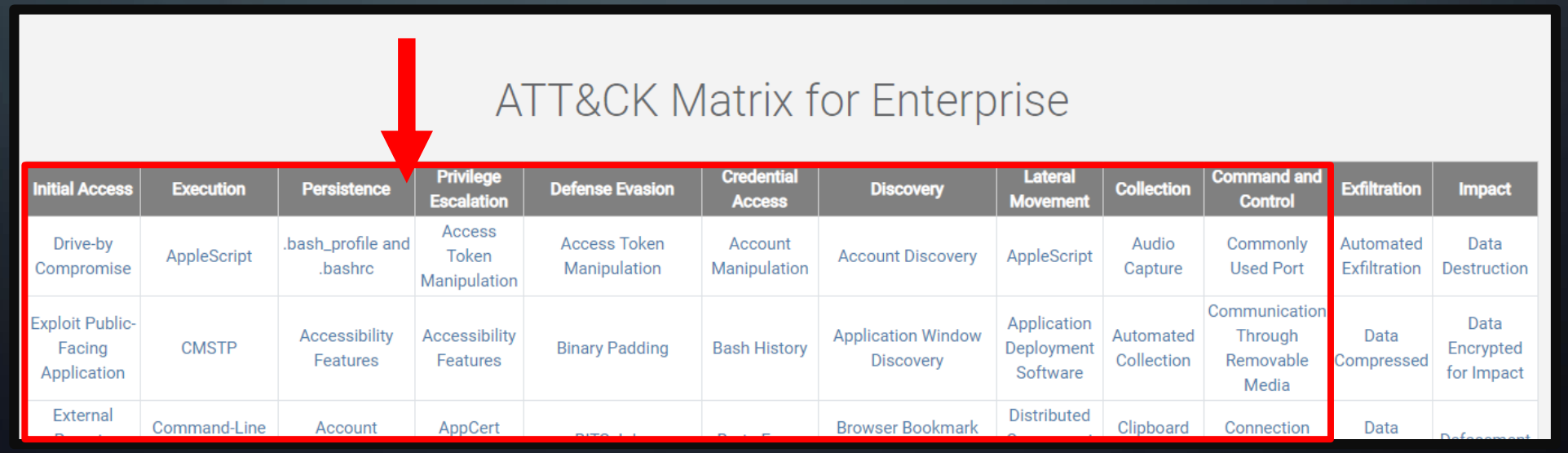

Clear history
Trojan:Win32/Zpevdo.B
Severe
23/9/2019
Trojan:Win
23/9/2019
See deta
Trojan:Win32/Zpevdo.B
Alert level: Severe
Status: Removed
Date: 23/9/2019
Recommended action: Remove threat now.
Category: Trojan
Details: This program is dangerous and executes commands from an attacker.
Learn more
Affected items:
file: C:\Users\JS\Downloads\RAT.zip
webfile: C:\Users\JS\Downloads\RAT.zip|https://raw.githubusercontent.con
pentestpartners/PTP-RAT/master/RAT.zip|
pid:14828,ProcessStart:132137126911361356
OK

DETECTION?

- IRL? Not that we know of, at least not publicly.
- Google searches suggested that the hype was in Nov 2017, and died in Nov 2017.
- Some suggestions from other professionals:
  - Detect a pattern of function calls
  - Machine Learning – detect abnormal pixels grouping / patterns

DEFENSE PERSPECTIVE

- Data Exfiltration is one of the last stages of an attack.
- Focusing strong preventive/detective controls in-place to tackle the previous intrusion phases is more meaningful.
- Like the perimeter, one does not expect 100% success.



ATT&CK Matrix for Enterprise

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Data Destruction
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Encrypted for Impact
External	Command-Line	Account	AppCert			Browser Bookmark	Distributed	Clipboard	Connection	Data	

IS IT USED IN THE WILD?

“If you're not looking for it, you probably won't see it.”

- Unknown


Evil pixels: Researcher demos data-theft over screen-share proto…
https://www.theregister.co.uk › 2017/11/09 › evil_pixels_researcher_demo...
Nov 9, 2017 - It's the kind of thinking you expect from someone who lives in a volcano lair: exfiltrating data from remote screen pixel values. The idea comes ...
Data exfil using pixel colour values, demo over RDP : netsec - Re…
https://www.reddit.com › netsec › comments › data_exfil_using_pixel_colou...
Nov 8, 2017 - 5 posts - 5 authors
Data exfil using pixel colour values, demo over RDP .... Surely there are more optimal ways to encode the data than straight pixel/data conversion. ... idea, but for just trying to exfiltrate a moderate amount of data, even the ...
pentestpartners/PTP-RAT: Exfiltrate data over screen interfaces - …
https://github.com › pentestpartners › PTP-RAT
Nov 13, 2017 - Exfiltrate data over screen interfaces. ... at https://www.pentestpartners.com/security-blog/exfiltration-by-encoding-data-in-pixel-colour-values/.
What is Data Exfiltration? Learn about Data Exfil | Forcepoint
https://www.forcepoint.com › cyber-edu › data-exfiltration
Data exfiltration is any unauthorized movement of data. It can also be known as data exfil, data exportation, data extrusion, data leakage and data theft. Whether ...
Missing: pixel | Must include: pixel
Data exfiltration tool PTP-RAT encodes data in pixel colour value…
https://ec2-35-168-102-112.compute-1.amazonaws.com › 2017/11/10 › data...
How to exfiltrate data from a machine that doesn't have file transfer capabilities or whose Remote Desktop Protocol (RDP) connection has been locked down, ...
How to Transfer Data via Monitor Pixel Color Values - GBHackers
https://gbhackers.com › transfer-data-via-monitor
May 5, 2019 - So, here we can Exfiltrate data via monitor pixel color values(Monitor Screen as Convert channel). Data Exfiltration Scenario: Attacker has ...
What is Data Exfiltration? | Digital Guardian
https://digitalguardian.com › blog › what-data-exfiltration
Sep 11, 2018 - Learn more about data exfiltration and methods for preventing data loss in Data Protection 101, our series on the fundamentals of data security.
Missing: pixel | Must include: pixel
Goooooooooogle

FUTURE WORK

- **DEFENSE**:
  - Preventive & Detective Controls
  - Include this in part of your Pentest / Incident Response / Reverse Malware Engineering process – it might just do this weird pixel thing.
- **OFFENSE**:
  - Come up with anti-compression techniques to increase accuracy. E.g. 100% accuracy allows reliable binary files transmission!
  - Port code into frameworks or to other script languages (e.g. powershell, Vbscripts, etc) to live-off-the-land (LOLscripts?)

QUESTIONS TIME!

RET;

[+] Program terminated.

[*] Thank you for listening.

[+] **Pixelcat** will be shared after this over github.com! ☺

REFERENCES

- Heavily-inspired by Pentest Partners' PTP-RAT:

https://www.pentestpartners.com/security-blog/exfiltration-by-encoding-data-in-pixel-colour-values/

- PTP-RAT Git hub repository:

https://github.com/pentestpartners/PTP-RAT