A Taxonomy of Encryption and Encoding Algorithms Used by Advanced Persistent Threats with Emphasis on Bespoke Encryption Algorithms

***Abstract* –** This paper reviews encryption, encoding and compression algorithms that have been used by Advanced Persistent Threats (APT) in their attacks on Microsoft Windows systems. These algorithms have been documented by the cyber security industry mainly in the form of white papers. The algorithms range from established international encryption standards to bespoke. The paper draws on Shannon's Law for the communications theory to support the discussion. The techniques and algorithms were analysed using C programs written for the purpose and spreadsheets. It concludes that most APTs use a level of encryption proportionate to the level of security needed but there are some misalignments with Shannon's Law.

Keywords –Microsoft Windows; Encrypt; Decrypt; Encode; Decode; Compression; Obfuscation; Advanced Persistent Threat (APT); Malware; Monte Carlo Simulation;

1. Introduction and Literature Review

The cyber security industry publishes the outcome of their analysis of Advanced Persistent Threats (APT) mainly in the form of white papers. Most of the documentation for this paper is based on two repositories of white papers, company provided information, webpages, blogs and some academic papers: the first repository is a corpus derived from web searches that were used as the basis for previous research; the second repository is of similar items from githib (Unknown, 2022). There is some overlap between these two repositories but in total there are over 3,300 documents. Analysis was restricted mainly to examples that contained high-level code, pseudo-code or a good written description. Some shorter assembler routines were analysed but time, complexity of larger assembler routines and report size meant that not all were.

The overlap between the two repositories creates problems in trying to compile meaningful statistics. Duplication exists within at least one of the repositories and across both. This was confirmed in the analysis for this paper as well as a review of papers based on file name. Additionally, different cyber security companies are contracted to different businesses. They may see the same attack infrastructure and report on it. The cyber security

companies also have different names for different APTs. Hence, what may appear to be different APTs using a similar attack is not.

Three independent advanced pdf searches on the aggregated repositories were performed using the using the keywords “crypt”, “encode” and “decode”. Search returns were then reviewed and analysis performed on the bespoke algorithms. The encode and decode searches highlighted the use of Basexx (where x is a two-digit number) encoding. Nine further searches were performed for “Base1”, “Base2”, … , “Base9”. A final search was performed for “compress” and “obfuscat”.

This paper notes the Classification of Encryption Methods (Singh, 2013, pp. 33-38), a malware survey (Gandotra, 2014, pp. 56-64) and a survey of Malware Obfuscation Techniques (You, 2010, pp. 297-300). There is little observed APT use of transposition algorithms.

Although APTs use a variety of algorithms, Shannon (Shannon, 1949, pp. 656 - 715) asserts that:

- The amount of secrecy should be proportionate to the effort put in to securing the message;
- The key size should be as small as possible;
- Complexity of enciphering and deciphering should minimised;
- Error propagation should be minimised;
- The encrypted message should be no longer than the message.

Some algorithms in this paper do not follow all of these rules but most seem to be strong enough for the level of secrecy and obfuscation desired. This report demonstrates that Base64 and any encoding less than 8-bit ASCII extends the encrypted message length from the original message and, therefore, does not align with Shannon’s fifth point.

Limitations were found in almost all of the bespoke algorithms. This paper suggests that this does not necessarily mean poor cryptographic knowledge by the malware authors as suggested by BlackBerry (BlackBerry, 2020, p. 23): “It was clear the original authors were not cryptography experts, … “. This paper asserts that there is no evidence of the underlying business philosophy of the different APTs and, therefore, what they are trying to achieve with

respect to the demonstration of their knowledge and Intellectual Property. Poor cryptographic knowledge is one explanation for some of the algorithms but it may be that the APTs have selected what they consider to be the best algorithms for the task in hand as asserted by Shannon's first point, above. Furthermore, they may not wish to demonstrate their full level of knowledge.

Mandiant state (Mandiant, 2020, pp. 23, 32) that in the year under report 13% of the malware was compressed, 4% encrypted and 49% of attacks had malware obfuscated/packed. The compression figure compares favourably to the 436 reports from the over 3,300 reviewed. The encryption figure of 15% differs from Mandiant's and 18% of the full repository contained a reference to obfuscation. It is noted the analysis for this report spans multiple years and Mandiant's report only one.

Cyber security companies may, for business reasons, not report all they observe. This paper provides examples of each type of bespoke algorithm reported, Examples that are close enough to those discussed in this paper are not included. Referencing every instance of encryption and encoding would make the paper intolerably long. This structure of this paper first lists internationally accepted algorithms, followed by bespoke algorithms. Comments on the bespoke algorithms are provided in each sub-section. To conclude the paper there is a short discussion and a critique.

Over 50 C programs were written in support of this paper to test the cryptography of various APT encryption algorithms.

This paper does not claim completeness of APT encryption algorithms but it is believed that it does provide a good insight into encryption algorithms and techniques used by APTs.

This paper is agnostic towards the origin and intent of APTs.

2. Use of Publicly Available Encryption Algorithms

To protect malware and data extraction APTs may use encryption or encoding Some of these encryption algorithms are publicly available such as:

- AES (Serper, 2020);
- RSA (Hromcová, 2019);

- RC4 (sKyWIper-Analysis-Team, 2012b, pp. 8, 35), RC5 (Symantec-Security-Response, 2014, p. 11), RC6 (Kaspersky, 2015c, pp. 27-28), RSA-2048 (Matthieu Faou, 2020, p. 18), Spritz (Avisa-Partners, 2020a, p. 18), (D. Huss, 2017, pp. 6-7);
- CAST (Symantec, 2015a, p. 13), Camellia (FireEye, 2014b, p. 7);
- Tiny Encryption Algorithm (TEA) (Fidelis, 2014, p. 4), XTEA, XXTEA (Kaspersky, 2015a, p. 18);
- DES (CrowdStrike, 2014b, p. 32), 3DES (Sofacy, 2015, p. 7);
- ElGamal (GovCERT.ch, 2016, p. 10);
- HC-128 (F. B. Perigaud, Boris, 2014);
- Blowfish (Levene, 2015, p. 4);
- Blowcrypt (proofpoint, 2015, p. 8);
- SALSA20 (GReAT, 2016b, p. 3);
- VEST (ptsecurity, 2021).
- RijndaelManaged (Ehrrlich, 2021, p. 15);
- OMEMO encryption and OTR encryption over XMPP (ZLAB, 2018, p. 4);
- an online PE crypter, Cassandra (Telsy, 2020, p. 4).

Others techniques are blends. For example, encryption using a hybrid encryption of Blowfish-OFB combined with RSA (Hromcová, 2019) or TripleDES followed by AES (Serper, 2020) or a combination of SALSA20 and Curve25519 (M-TRENDS, 2021, p. 24).

Use is also made of Windows and other software:

- CryptProtectData (Kaspersky, 2015e, p. 6), TrueCrypt (Symnatec, 2015, p. 13) and CryptEncrypt/CryptDecrypt (Elastic, 2020), (Hromcová, 2020, p. 7);

- SSL (ClearSky-Cyber-Security-ltd, 2021, p. 21), TLS (Hegel, 2018, p. 6). The related ChaCha stream cipher has also been seen (FireEye, 2019c), OpenSSL (eset, 2019b, p. 19);
- the certutil utility (Breitenbacher, 2020, p. 6);
- BitLocker (profero, 2021, p. 3);
- BatchEncryption (Jazi, 2021, p. 15);
- ConfuserEx (GReAT, 2021), (Gorelik, 2019, p. 3);
- Open source NXCrypt (eset, 2019d, p. 10);
- publicly available Ransomware-as-a-Service (RaaS) Encryptor RaaS (FireEye, 2019a, p. 48);
- JSEncrypt (welivesecurity, 2021);
- Number Theory Library (NTL) (BAe-Systems, 2014a, pp. 10, 25);
- IonCube (Huss, 2021, p. 41);
- UserForm1 encoding (McAfee-Labs, 2020);
- WolfCrypt (Matthieu Faou, 2020, p. 12).

The scope of this paper is Microsoft Windows hosted malware but it notes Apple hosted malware: CCCrypt (ti.qianxin.com, 2019), Apple's CommonCryptor library (BLACKBERRY-RESEARCH-&-INTELLIGENCE-TEAM, 2020, p. 39).

Although strictly not encryptors, use has been made of CryptStringToBinaryA (Cyber-Geeks, 2021, p. 21), Aaencode for obfuscation (Rostovcev, 2020, p. 39) and Allatori a Java obfuscator (Singh, 2020).

3. Use of Publicly Available Compression Algorithms

Observed compression algorithms are:

- CAB (Unknown, 2010, p. 14);
- LZip (Command-Five-Pty-Ltd, 2011, p. 2);

- LZ77 (Nart; Wilhoit Villeneuve, Kyle 2013, p. 10), a custom Lempel-Ziv-based algorithm (Global-Research-and-Analysis-Team, 2013, p. 10);
- EZIRIZ (Fidelis-Cybersecurity-Solutions, 2013, p. 2);
- LZO fast compression (RSA, 2014c, p. 27), bzip2 (Kaspersky, 2014b, p. 22), LZO (Novetta, 2014a, p. 13), LZO1X (Novetta, 2014c, p. 7);
- modified LZMA-compression (Fagerland, 2014, p. 19), Gzip (Antiy-CERT, 2015, p. 3); Flex-compressed .SFX file (C. S. Pernet, Eyal, 2015, p. 23);
- multiple compression techniques in one attack, LZJB, LZF, FastLZ, LZO, (Kaspersky, 2015b, p. 18) and observed over a period (Cisco-Talos-Intelligence-Group, 2021) ;
- LZ4 (Hiroaki, 2021, p. 32);
- ZWS (Bryan ; Grunzweig Lee, Josh 2015, p. 1);
- lzma (Symantec, 2014, p. 12), 7-zip (Alperovitch, 2014, p. 5);
- LZ Huffman compression algorithm (lzhuf) (Settle, 2016b, p. 24);
- LZSS (CHEREPANOV, 2016, p. 20);
- LZHAM (Cylance, 2018, p. 30);
- QuickLZ (Doctor-Web, 2020a, p. 25);
- recursively using GZip (Lifshitz, 2021, p. 10);
- ZLIB/DEFLATE (Command-Five-Pty-Ltd, 2012b, p. 2), ZLIB (Spohn, 2012, p. 23). .7Z (Chang, 2015, p. 6), ExOleObjStgCompressedAtom (Helios-Team, 2016b, p. 24);
- PPMd format (sKyWIper-Analysis-Team, 2012a, pp. 8, 35);
- UPX (Cox, 2012, p. 11);
- RAR (Mandiant, 2013, p. 37), modified RAR software (CyCraft-Research-Team, 2020, p. 4), WinRAR (GReAT, 2019, p. 5);

- RtlDecompressBuffer API (LZNT1). including UCL compression (Raiu, 2013, pp. 5-6);
- NRV2e algorithm from the open-source UCL library (Kaspersky, 2013b, p. 7), Compression library based on Nrv2d / UCL (securelist, 2015);
- An LOFDM System Peak to Average Ration Non-Linear Compression and Expansion Algorithm (ThreatConnect-Inc.-and-Defense-Group-Inc., 2015, p. 77);
- zip or lzh (S. N. Tomonaga, Yuu, 2015, p. 11);
- zlib, libbz2, and ppmd (Scott, 2016, p. 52);
- a "very rare compression algorithm" (Helios-Team, 2016a, p. 32) (Q.v. ExOleObjStgCompressedAtom above);
- .BZ, .ACE (The-Cylance-Team, 2016, p. 3);
- Jcalg1 (Global-Research-and-Analysis-Team, 2016, p. 3);
- a custom implementation of the Lempel–Ziv–Welch (LZW) algorithm (eset, 2016, p. 73);
- aPLib, custom implemented LZW, LZM, zLibalgorithm (Draco-Team, 2020, pp. 6, 7, 8, 14, 15);
- jpeg compression (Travers, 2017, p. 25);
- the legitimate tool "mpress.exe" (Bar, 2017);
- ZPP (Trend-Micro, 2017, pp. 3, 35-36);
- COMPRESS (Falcone, 2018);
- unpack200.exe decompression for Java 8 (Co-Authored-by-Rapid7, 2019, pp. 16-17);
- exec,powershell -Command compress-archive (Cyber-Safety-Solutions-Team, 2019);
- system tar command (netlab.360.com, 2020);

- Minedoor (ANSSI, 2020, p. 10);
- the legitimate compression utility XZ Utils (Expert; Park, 2020, p. 2);
- Webp.GetFrame() method (Microsoft, 2021).

There is a example of a small bespoke compression algorithm APT (us-cert-cisa, 2020, p. 7) where two bytes compressed into one there is no explanation of how this might be reversed, while another (amnesty.org, 2020, p. 8) uses a custom compressor which is not further elaborated on.

4. Use of Base 16, 32, 64 etc. Encoding

Base64 was, by far, the most commonly used form of BaseXY encoding, where X and Y are integers. A search for the string "Base64" revealed that approximately 22% of the papers presented this method of encoding. However, this may not be representative. For each of the two repositories, the percentages were about 14% and 30%. These three percentages provide an indication of the range for the number of attacks using Base64 encoding.

Examples of non-Base64 codes are:

- Base16 (F. Perigaud, 2014a, p. 6) where each nibble is encoded using the characters A to P;
- Base25 (welivesecurity, 2020a, p. 13) where the Base25 decode is used with key subtraction – the code is helpfully provided;
- Base26 (BAe-Systems, 2014b, p. 3);
- Base32 (Palo-Alto-Networks-Blog, 2016), and (eset, 2020, p. 20) where a modified base32 encoding, with a custom conversion table has been seen;
- Base36 (Singh, 2016) where a Base36 random number is used;
- Base52 (Horejsi, 2018) where a macro has been seen with strings encoded in base52. Interestingly, one week after the original paper was published in November 2018 the APT changed from Base52 to Bases 40, 45, and 48 (Horejsi, 2018);
- Base85 (Chen, 2021) where a shellcode is base85 and hex-encoded;
- Base91 (securityintelligence.com, 2019);

- Base128 (QuoIntelligence, 2020, p. 5).

The use of Base64 XORed with a single hex key is noted (zscaler, 2011, pp. 6-7): "... that have been base64 encoded and then XORed. XOR keys of 0x3Cand 0x3E have been observed.". There are a number of examples e.g (Moran, 2014, p. 3) (The-SecDev-Group, 2009, p. 37), of use of Base64 encoding enciphered (XORed) with values that, at the point of publication, are not identified.

Some APTs use a non- canonical (custom) alphabet (FireEye, 2014c) (Moran, 2014, p. 2), (van Dantzig, 2015, pp. 11, 29), (Denbow, 2012, p. 7), (Bar, 2016).

Another encodes the data, by byte reversing, encoded with base64, and reversed again (Check-Point, 2015, p. 10).

Anything less than Base256 encoding for 8-bit ASCII encoding does not satisfy Shannon's fifth point as the encrypted message is longer than the message. For example, in Base64 3 8-bit ASCII characters (24 bits) are encoded to 4 6-bit Base64 characters. The example, below, illustrates how Base64 encoding converts the first three octets of a PE file into four encoded characters and XORed with 0x3C:

Source	**Text**	M (0x4D)								Z (0x5A)								0x90							
Bits		0	1	0	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0	0	1	0	0	0	0
Base64 coded	**Sextets**	19						21						42				16							
	Character	T						V						q				Q							
	Octets	0x54						0x56						0x71				0x51							
	XOR 0x3C =	0x68						0x6A						04D				0x6D							

Table 1: Selected List of APT Developed Single Key Encryption Techniques

It is clear that all ASCII characters with most significant bits of 010011 (L, M, N, O) will Base64 encode to T. Likewise, all ASCII characters least significant bits of 010000 (0x10, P, 0x90, 0xD0) will Base64 encode to Q. Similar analysis may be performed on the other two characters. A post-Base64 encode with XOR of 0x3C or 0x3E (note: the two most significant bits are zero in each case) means each tetrad encodes to another tetrad of

characters. It was hypothesised that this would mean there is an excess of equal characters at an offset of four, eight etc. in a base64 encoded file. This was tested what might be the type of document that an APT might download.

A Microsoft Word document was downloaded from the University of Gloucestershire's website and tested at offsets 1 to 19. The 15818-long file would expect 2472 hits at random (it is noted that text is not random). A range of counts were seen, ranging from low 2,000s to low 3,000s except for multiple of four offsets where the counts were about 10 times as high.

A PE file contains alot of 0x00 characters which makes it easy to strip off the characters used in any XOR operation. Identification of this technique from other files will show excess frequency counts for every four postilions. The outcome of an XOR is a simple substitution and is the equivalent of an XOR on every Base64 element changing the canonical alphabet to a new alphabet.

The above test works for APTs which use a non-canonical base64 alphabet.

It is noted that any BaseXY encoded document will be restricted to XY unique characters, which is another test for BaseXY.

Double encoded base64 has been observed (telsy, 2019), (McAfee-Labs, 2020), the latter for a DLL implant. Base64 increases the file length by a third so a double Base64 would increase the file size by over 77%. This does not comply with Shannon's Law.

5. Bespoke Algorithms

This paper will now review bespoke algorithms. Nearly all are shown to contain weaknesses but their implementation may be deliberate. It is asserted that a good APT will use the level of attack and encryption suitable for the target and also suitable for the appearance that the APT wishes to demonstrate, if discovered.

5.1. Caesar Cipher

Although not necessarily bespoke, the Caesar cipher is used in different ways. The Caesar cipher encodes by adding the same character to each letter of the text. For a 26 letter Latin alphabet. This may be as simple as adding 1 (modulo 26): A becomes B, B becomes C, …, Z becomes A. For a 256-character, 8-bit ASCII bitwise Exclusive OR (XOR) may be used.

The table below summarises some straightforward encryption schemes:

Encryption Algorithm	Key	White Paper Reference
XOR Repeating	0x66	(Trend-Micro, 2012, p. 13)
XOR Repeating	0x02	(Dela Paz, 2012, p. 6)
XOR Repeating	0x90 (with a 16-byte key also being used)	(Alintanahin, 2015, pp. 3, 7)
XOR Repeating	"1/2"	(C. S. Pernet, Eyal 2015, p. 16)
Multiplication	One-byte key	(Alintanahin, 2015, pp. 3, 7)
Unknown	Machine specific variables e.g. MAC address	(Villeneuve, 2012a, p. 5)
Double encryption Repeating	0x2C and 0x7B 0x70 and 0x79	(Fidelis-Cybersecurity-Solutions, 2015, pp. 5, 7)

Table 2: Selected List of APT Developed Single Key Encryption Techniques

However, Schneier (Schneier, 2007, pp. 10-11), states that a simple substitution can be easily broken. and goes on to describe how.

At least two APTs (Haq, 2014b, p. 21) (Cylance, 2018, p. 30) perform a bitwise byte inversion (NOT) for encryption. Another (Kaspersky, 2014a, p. 99) reverses the plain text and then XORs a single byte against every four bytes. One APT (eset, 2019c, p. 13) uses ROT13.

5.2. Lookup Table and Equivalents

There are examples of ROR (rotate right shift) or ROL (rotate left shift). It is noted that a ROR of x bits on a 8-bit byte is the equivalent of a ROL of (8-x) bits.

One APT (GReAT, 2018a, p. 5) encodes text with 0x40 and then performs an ROR of six bits. This is equivalent to a simple substitution. The 0x40 only inverts one bit and after the ROR that bit is the least significant bit. Another (B. G. Levene, Josh; Ash, Brittany 2018) performs an addition and then ROL of three bits. Again, this is equivalent to a simple substitution.

This use of ROL to left shift based on the encrypted character's position within the text is seen elsewhere (TIVADAR, 2013b, p. 6): the text length is less than 1024 and each character is ROL'ed the text length minus its position times; This has the effect that every eight characters is plain text. As an aside, this same APT (TIVADAR, 2013a, pp. 10-12) performs other bespoke obfuscation: "… the malware does not use the same hash for encryption. Instead, it would interchange the first DWORD with the second one in the structure and would re-compute the SHA-1 hash".

One decryption algorithm (Crowdstrike-Global-Intelligence-Team, 2015, p. 9) which is presented by the Python code:

chr(((ord(x)^(0x1C +1)) + (0x1C +1)) & 0xFF)

A program was written to generate key for all values of x [0, 255]. These "x and key" pairs were analysed in a spreadsheet pivot table and found to be a 1-1 mapping i.e., a simple substitution or a one-character codebook of size 256. There is a lot of structure is observed in the codebook when written 16 digits to a line:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	58	57	60	59	54	53	56	55	50	49	52	51	46	45	48	47
1	42	41	44	43	38	37	40	39	34	33	36	35	30	29	32	31
2	90	89	92	91	86	85	88	87	82	81	84	83	78	77	80	79
3	74	73	76	75	70	69	72	71	66	65	68	67	62	61	64	63
4	122	121	124	123	118	117	120	119	114	113	116	115	110	109	112	111
5	106	105	108	107	102	101	104	103	98	97	100	99	94	93	96	95
6	154	153	156	155	150	149	152	151	146	145	148	147	142	141	144	143
7	138	137	140	139	134	133	136	135	130	129	132	131	126	125	128	127
8	186	185	188	187	182	181	184	183	178	177	180	179	174	173	176	175
9	170	169	172	171	166	165	168	167	162	161	164	163	158	157	160	159
10	218	217	220	219	214	213	216	215	210	209	212	211	206	205	208	207
11	202	201	204	203	198	197	200	199	194	193	196	195	190	189	192	191
12	250	249	252	251	246	245	248	247	242	241	244	243	238	237	240	239
13	234	233	236	235	230	229	232	231	226	225	228	227	222	221	224	223
14	26	25	28	27	22	21	24	23	18	17	20	19	14	13	16	15
15	10	9	12	11	6	5	8	7	2	1	4	3	254	253	0	255

Table 3: Mapping

Note the repeating difference between rows (16, -48, 16, -48 …), the repeating difference between columns: (1, -3, 1, 5, 1, -3 …) and the repeating difference at the end of

one column to the beginning of the next (5, -59, 5, -59). All three sets of differences are modulo 256.

There are other similar algorithms (Paul, 2013, p. 36), including two different Python encoded algorithms (Guarnieri 2014, pp. 9-11), which have an encode and decode and, again, are equivalent to a simple substitution.

Bitwise inversion has already been discussed but this is extended (RSA, 2014b, p. 19) where the beacon is enciphered with a byte XOR with 0x5f followed by an bit-wise inversion of each byte. The is equivalent to taking the original text and XORing with 0xa0 i.e. the bitwise inverse of 0x5f.

As part of encoding one arithmetic addition, XOR and shifting is seen (Settle, 2016a, p. 31). Again, this is a simple substitution.

Elsewhere is seen (Shevchenko, 2017) an XOR decryption:

```
buffer[i] ^= 0xCC ^ ((buffer[i] ^ 0xCC) >> 4);
```

This has the effect of filling the most significant nibble with 0xC and the least significant nibble with the most significant nibble of buffer[i]. For example. Assume that buffer[i] is 10000001. An XOR of 0xCC gives 01001101, right shift by 4 gives 00001000 and XOR with 0xCC gives 11001000 i.e. the most significant nibble of the original text is now 0xC and the least significant nibble is the previous most significant nibble. These eight bits are then used as the key. However, this code suggests that to decrypt the text the enciphered text is needed i.e. buffer[i].

Another (Dr.Web, 2020b) generates key from a CRC32 table.

An encryption algorithm using arithmetic and XOR had been observed (TIVADAR, 2013a, pp. 10-12). For each 0x40 (decimal 64) positions of plain text a counter (initialised to zero) is decremented, and then XORd to the plain message. The counter is then added to this intermediate step. Both steps are a 1-1 mapping and again the “double encryption“ has the effect of single encryption produced by a Vigenère Square. A program was written to implement the encryption algorithm and produce a table for all 256 8-bit ASCII characters for each position of plain text. In a similar manner to the previous section, much structure was observed in the Vigenère Square. The most striking effect is the encryption of 255 (binary

11111111, 0xFF). Any ROR of this number leaves the number the same. An XOR of any number with 0xFF is a binary inversion of the original number. Which is a mod 256 complement of the number. When added back in the result is 256 e.g., 229 11100101 when XOrd with 0xFF gives 00011010, decimal 26. Added to 229 is 255. The encryption algorithm decrypts twitter links so 255 should never be seen but it provides an insight into a limitation of the algorithm.

Another APT (Bryan; Grunzweig Lee, Josh 2015) performs three 1-1 map using subtraction, XOR and addition. Again, this is equivalent to a 1-1 lookup table which is highly structured. While one (Novetta, 2016b, pp. 31-32) uses addition and then XOR, and XOR then subtraction.

One APT (DATA-SecurityLabs, 2014, pp. 7-10) appears to use two different encipherments in the same attack: the malware obfuscation layer ROR by six bits, while elsewhere a driver which decodes a file uses a XOR of 0x73 and then subtracts 0x57. Other examples include (Bryan; Grunzweig Lee, Josh 2015). Another performs an ROR/ROL of four position after zlib compression and before XOR with 0x23 (FireEye, 2015, p. 55). One author (Gross, 2015, p. 2) notes that 4-byte XOR which is permuted using a byte rotation. The author notes that this repeats after 256 bytes and this was conformed by analysis.

There is an example of different combining mechanism (Klijnsma, 2016, p. 34): the text is encoded using XOR of key if the position in the text is odd and adds if the position is even. A similar algorithm is seen (Gross, 2016, p. 5) where use is made of a single byte XOR but the algorithm skips the zero byte and the key itself.

Elsewhere (Bitdefender, 2016, pp. 4, 16) combination of a static value and two positional counters is seen. This has the effect of a producing highly structured key where every other key value alternates between 250 and 251 and the other alternates have constant differences. This same APT later uses a similar encryption algorithm which produces even greater structure. In the first example, the structure comes partly from the combining method, part of which uses a constant. Key at the next position is a function of two XORs of key of the previous position. This has the effect of the constant being applied at one position to be stripped off at the next. In the second example one of the XOR elements is 200 + i (the positional counter). Again the 200 is XOR'd in and out every two key characters.

One algorithm (welivesecurity, 2020b, p. 4) produced key which was limited in the last four positions when viewed as a decimal number. A program developed for analysis modified the sample code changing the "+" operator to "^" and then "&". The first manifested key limited in a similar way to "+" while the latter produced every key as decimal 17.

Similar encryption algorithms are seen across attacks and reported on by different companies within attack (Dr.Web, 2020c), (Insikt-Group, 2020, p. 9), (F. Perigaud, 2014b).

One (Mercer, 2020) encrypts the low order nibble with the high order nibble: byte = byte XOR (byte >> 4).

Jscript inside a malicious SCT file has been discussed (Wueest, 2017). This is, in effect, a simple form of obfuscation. It is also encoding but it does meet the one of Shannon's Principles in that the encoded stream is longer than the original. This code "\x52\x32\x56\x30" etc. A similar algorithm is used elsewhere (MalwarebytesLABS, 2018) who observe script obfuscated using hex code, for example "var _0x8aa6=["\x75\x73\x65 …."

5.3.Double Encryption

Fidelis-Cybersecurity has two examples of XOR double encryption (0x2C with 0x7B) and (0x70 with 0x79) (Fidelis-Cybersecurity-Solutions, 2015, pp. 5, 7). In mathematical terms, the XOR Boolean operator is a GF(2) Finite Field and therefore exhibits the properties of commutativity and associativity – no matter which way the operations are performed the answer is always the same e.g. Using the two bytes 0x2C with 0x7B from above to encrypt with XOR the lower-case letter "a" (0x61) it can be seen that this is equivalent to using a single byte for encryption:

(0x2C XOR 0x7B) XOR 0x61 = 0x57 XOR 0x61 = 0x36

(0x7B XOR 0x61) XOR 0x2C = 0x1A XOR 0x2C = 0x36

0x7B XOR (0x2C XOR 0x61) = 0x7B XOR 4D = 0x36

Similarly (Qihoo-360-Technology, 2018) there is encryption by adding 0x34 and XORing 0xA4. Again, this is a 1-1 mapping and then another 1-1 mapping i.e. a simple substitution.

Many of the bespoke “Double Encryption” algorithms reduce to a simple substation offering less protection.

5.4.Finite Repeating Key

Schneier (Schneier, 2007, pp. 13-15) states that there is no real security here and again goes on to describe how it is broken.

The table, below, summarises some repeating encryption schemes:

Encryption Algorithm	Key	White Paper Reference
Unknown Repeating	16-byte key	(CrowdStrike, 2014a, p. 27)
XOR Repeating	4-byte key	(Wyke, 2011, p. 8), (RSA, 2014a, pp. 13, 26-27)
XOR Repeating after bitwise NOT of the input stream.	4-long key’\x30\x30\x34\x31’ (said to be random)	(Haq, 2014a, p. 13)
XOR Repeating	172-byte	(Falcone, 2017)

Table 4: Selected List of APT Developed Repeating Key Encryption Techniques

One APT (D. M. Huss, Matthew, 2017, pp. 6-7) uses a repeating 38-byte key, to which is added an incremental counter and then XORd with 4-bytes to produce, in effect, a 256-long key.

Not all counters start at zero: One (Hwa, 2014) uses a counter based XOR with the key starting at 66. A program written to trivially simulate this algorithm demonstrated the theory that that key repeats after 256 iterations. It is possible that the technique is used on other attacks by the same APT each with a different start for the counter.

Another (Marschalek, 2014, p. 8) does a seven-bit rotation and an XOR of an 8-digit hex key. Elsewhere (Soo, 2017) key is created by taking the eight least significant bits from a bitwise inverse of a right shift of 0x6121 controlled by an integer counter mod 32. This has the effect of producing a repeating 32-long byte stream.

One APT (Trend-Micro, 2019) specifically mentions a Vigenère cipher as its method of encipherment.

It can be seen that some repeating key streams obviously repeat i.e. the stream is a fixed value in a variable. Others non-obviously repeat due to the structure of the algorithm. It is not known if the respective designers are aware of the latter effect.

5.5. Finite Non-Repeating Key

One APT (RSA-RESEARCH, 2017, pp. 23-24) encrypts using an XOR of an MD5 encoded password, then XORs with a fixed byte. The latter is presumably to obfuscate the MD5 algorithm and deny any attempts to exploit its vulnerabilities. Base64 encoded is then applied.

Another APT (ESET-Research-Whitepapers, 2018, p. 17) takes a seed which is passed to srand. Further calls to rand generate the key to be XORd with the text. This paper notes that the same seed passed to srand will always produce the same key stream produced by rand. This allows the attackers to produce unique key streams across their attack portfolio but the same key stream within any of those attacks.

One APT produces key that has a difference of 17 and zero every four key production characters when the main loop variable (a5) is initialised to zero (Lunghi, 2020, p. 16).

5.6. Positional Encoding.

This paper defines Positional Encoding as any algorithm that in order to encode the data, each character has its positional number added to, or subtracted from, it. e.g. the nth position has a number |”n” added or subtracted to it in a manner describe by FireEye (N. B. Villeneuve, James T. ; Moran, Ned; Haq, Thoufique; Scott, Mike; Geers, Kenneth 2013, p. 11).

An encoded IP address (Command-Five-Pty-Ltd, 2012a, p. 5) and control port is decoded by subtracting a single key for all positions plus the position modulo 8. This is equivalent to subtracting a repeating 8-byte key.

Another APT (Cutler, 2012, p. 3) encodes a small amount of information by adding each character's ASCII value to its offset from the start of the stream. There are no details given as to what happens if the result is greater than 255 but if the text to be encoded is short enough then this probably never happens. Should short, encrypted streams be seen this, or variants of it, then it could be easily tested.

One (Raiu, 2014, p. 3) adds the position to 0xa (i.e. decimal 10). This has the effect of adding 10 to every enciphering position value so directly hiding that positional value. This paper postulates that different starting constants could be used in different attacks by the same APT. Possibly an unique serial number for each attack.

Another APT (Palumbo, 2014, pp. 9-10) XOR encrypts in 8-byte rounds. For the final encipherment it uses a provided key stream XOR'd with a position counter. The values of the key stream are not provided in the documentation but encipherment starts at the end of the text and works forwards. This technique helps thwart analysis as enciphered texts are usually aligned and analysed from the start. Use of a positional counter for the sub-key will ensure different final key values for different plain text lengths. Only texts with the same length will be enciphered with the same final key. However, as the APT uses the lower byte of the word any file length 256 positions away will use the same sub-key value. This was demonstrated programmatically for file lengths 1000 and 1512.

One APT (Benchea, 2015, p. 20) passes a constant to the encryption routine which ensures that the encipherment routine can be re-used within the same attack. Another APT (Dr.Web, 2020a) takes the position off 32 to generate the key.

Another APT (Lee, 2018) uses a fixed 20-long key stream, steps through this and combines it with an inner count to produce key. A program was written for message length of 256 and key analysed/ It was rough with a chi-squared of are 6.6×10^{-153}. The message length was extended to 4096 and the key frequency table sorted by frequency, high to low and then key value. A difference of successive key value shows strong structure with the highest frequency (80) each appearing 15 times. A difference of the key values showed that they were 256 apart. Similar structure was observed in the key value frequency table.

This paper asserts that positional encipherment offers little security beyond obfuscation. Consider an IPv4 address positional encoded with the position and a single byte XORd across all positions. This may be solved by brute force: we simply XOR off all 256 possible byte values for 256 intermediate texts and subtract a sequence of numbers from each. The text that has only 0-9 and "." as the plain is a likely candidate for true plain text I.e. the IPv4 address. In addition, the key is 256 long and each value in the range [0, 255] appears once and only once. Assume 8-bit byte output then any addition greater than 256 (and subsequent mask) will repeat in the lower order bits. This was demonstrated programmatically for positions 0 to 4095 and an XOR of 0xAA (alternate 0s and ones).

5.7.Transposition

There is little reported use of bespoke transposition.

The first (FireEye, 2019b, p. 59) uses a remote administration tool designed as a .dll. The strings are encoded using a transposition algorithm which is not elaborated on by in the paper. The second (Microsoft-Security, 2021) "... uses a simple byte-swap decoding algorithm …" i.e. every two bytes are positional exchanged. A similar scheme is seen elsewhere (GReAT, 2016a) where use of a 1-byte XOR id followed by "replacement of the Odd byte for an Even byte in several hundred bytes from the header".

5.8.Steganography

At least one APT (Faou, 2019a, p. 21) makes use of steganography to hide a payload but, again, this breaches Shannon's Law. At least one APT (Brumaghin, 2019) has a PE hidden in a bitmap.

5.9.Autokey

The use of uses the XOR and additive for encryption has been seen (F. B. Perigaud, Boris, 2014) but the additive is the previous value of the plain text with the result that it is an autokey cipher. This is part of four-step encryption algorithm.

One APT (Novetta, 2014b, p. 2) decodes by starting at the last byte and XORing each previous byte with the current byte value in reverse order. This has been observed elsewhere (Trend-Micro, 2018).

Another APT XORs the encrypted character from the previous stage XORd with a fixed byte (0x15) as key (FALCONE, 2015, pp. 13, 17).

Differencing adjacent plain text values in a stream has been noted (Nart ; Wilhoit Villeneuve, Kyle 2013, p. 5). The first value is not actioned. Subsequent plain values are XORed with the previous value in the stream. The outcome here is that, statistically, a count of the first value of the stream will be the distribution of the unenciphered text and subsequent values with have a unenciphered difference distribution. Another (Unit-42, 2017) XORs plain text two positions apart.

5.10. Stream Cipher

One APT (M.Léveillé, 2019, p. 18) decrypts cipher using continuous key generated from a seed that depends on the first two cipher position. It is inferred that these two positions are key needed to encipher otherwise how could one encrypt when the encrypted data is needed to seed the algorithm? The key produced did not repeat for at least 4096 positions, the limit of the loop set in the program written for analysis.

Another APT (Doctor-Web, 2020b, pp. 45-46, 74) generates a stream key which exhibited no obvious non-random features for initial values [0-9] each generating 4096 keys. The exception was that each key value appeared once only. However, code later in the document produces key with a lot of structure exhibited. The structure appears both when BYTE1 an BYTE2 are interpreted as the lowest and second lowest bytes and also as the second and third lowest.

Elsewhere (Avisa-Partners, 2020b, p. 17) there appears to be a strong tendency for the key at the next position to be mainly comprised of the key right shifted 8 bits with new values at the left most 8 bits. The coalescence to this feature is very early in the key generation mechanism. This may be due to the use of left and right shifts in the key generator as well use of the Boolean AND operator.

5.11. "YHCRA" Encryption

One APT (N. d. T. Villeneuve, Jessa 2013, p. 5) has been seen using an encryption scheme where each byte is XOR-ed by every letter in the string, YHCRA, and rotated three bits to the right after every XOR operation. One interpretation of the algorithm is the equivalent to encrypting using the single byte 11110110 as shown in the table below:

Initial Text	(Null) 00000000	P (01010000)
XOR Y (01011001)	01011001	00001001
ROR Shift 3	00101011	00100001
XOR H (01001000)	01100011	01101001
ROR Shift 3	01101100	00101101
XOR C (01000011)	00101111	01101110

ROR Shift 3	11100101	11001101
XOR R (01010010)	10110111	10011111
ROR Shift 3	11110110	11110011
XOR A (01000001)	10110111	01000001
ROR Shift 3	11110110	01010110

Table 5: Analysis of "YHCRA" Encryption Scheme, Method 1

It can be seen that 11110110 XOR P (01010000) equals 01010110 which is the final value of the rightmost column.

There are two other interpretations: the first is that each plain text character is XORd with all of the YHCRA characters which are all ROR by 3. This would give a repeating key length of 8; the second interpretation is that the plain text is encrypted as described in method 1 and the process continues, without resetting, for the next plain text character. A program was written to emulate this and uncovered a 16-long repeating key.

Whichever of the three algorithms is correct, this example demonstrates that one must be careful with encryption as "more complicated" or "more complex" does not necessarily mean better, i.e. stronger, encryption. This example re-enforces Shannon's assertions above about the need to minimise complexity: in Method 1 the designer of the above scheme has picked a bit shift of three which is co-prime to eight but has five times the number of XORs needed plus five rotate shifts for each iteration i.e., a workload of at least five times for each encipherment when all that is needed is to store 11110110 as the single byte key. Depending on the hardware and software implementation the cost could be as much as 10 times (10 clock cycles) as much as using one byte.

5.12. Random Number Generators

A Linear Congruential Generator (LCG) (Schneier, 2007) is of the form:

$X_i = (Ax_{i-1} + B) \bmod m$

AN LCG may be used as part of a wider encryption algorithm, for example an n-stage encryption (Co-Authored-by-Rapid7, 2019, pp. 16-17). Here there is a rolling XOR encryption, followed by RC4 encryption of this stream, followed by Salsa20 encryption.

Stage 1 (the rolling XOR encryption) is a type of LCG with seeds of four and eight and a divisor of 255; i.e.

$x_{i+2} = (x_{i+0} + x_{i+1}) \bmod 255;$

$x_0 = 4; x_1 = 8;$

Analysis shows that this stream repeats after about 360 iterations. The sequence is:

$x_0 = 4; x_1 = 8; x_2 = 12; x_3 = 20; \ldots$

$\ldots x_{356} = 251; x_{357} = 4; x_{358} = 0; x_{359} = 4; x_{360} = 4; x_{361} = 8;$

It is possible that the developers are aware of the short comings of RC4 and used this technique to overcome them.

Another APT (Symantec, 2015b, p. 10) uses an LCG directly for encryption. The authors of the paper state that their organisation has not previously seen an LCG used for communications encryption.

A pseudo LCG has also been observed (Unit-42, 2017). What appears to be a single byte key and key offset are provided to a routine. Successive values of seed are generated:

seed = (seed + key_offset) % 0x100

Depending on the value of key_offset different values from different cycles are used. For example: for key_offset equal to 1, all 256 8-byte values are stepped through; for key_offset equal to 2, one of two 128-long cycles is used etc. Again, it is not known if the designers of this LCG are aware of the limitation and that there is no full cycle and only sub-cycles.

Another use of srand has been seen (FALCONE, 2015, pp. 13, 15, 17). It is seeded with a fixed seed and successive values from the call are reduced mod 128 with this result being used as an XOR key. It is noted that reduction modulo 128 only provides a 7-bit key in an 8-bit variable. A Python example demonstrates of cdll.msvcrt.rand() modulo 128 as the byte wise XOR key. This random number generator is initialised using cdll.msvcrt.srand(2014) to ensure that the key is the same each time. A probably related APT (Falcone, 2016) uses srand and rand but the seed is 2563. Elsewhere srand(time(0)) has been observed (Novetta, 2016b, pp. 31-32).

One APT (Group-IB, 2018, p. 51) uses rand but there is no indication that srand() has been used for initialisation.

The use of Mersenne Twister random number generator has been observed (eset, 2019a, p. 16). This implementation is a variant with unique seed for each machine stored in the machine's Registry.

The use of BCryptGenRandom (Tikhonova, 2021) is noted.

In another attacks (GReAT, 2018d, p. 2) log files are encrypted with a AMPRNG based custom encryption algorithm but no further details are given.

One APT (Nart; Wilhoit Villeneuve, Kyle 2013, p. 10) uses large portions of Makoto Matsumoto and Takuji Nishimura's Random Number Generator for encryption functionality.

One description (JPCERT-CC, 2016) of the Python rand() function uses an encryption seed which is twice the "config_offest", while another description states that "All strings inside a driver used by one are encrypted by XOR with a pre-seeded random number generator" (securelist, 2015). There is no elaboration on this. The use of srand seeded by the compilation timestamp is noted (Faou, 2019b, p. 10).

The selection of a random number generator should be done lightly: "Random numbers should not be generated with a method chosen at random" (Knuth, 1981, p. 5).

5.13. Symmetric Key Generation

One APT (S. Tomonaga, 2015, pp. 2-3) uses two 32-bit words to generate key. The registers are initialised and modified during the key generation process by four constants: the first appears to be a date and the other three are prime numbers which, of course, are co-prime to the 32-bit words. The analysts note that these values may vary, presumably across attacks. Key is generated by taking all eight non-overlapping bytes from each word and combing them with a mixture of XOR and subtraction. It is possible that there is a typographic error as one of the combining variables is given as the full word. There is only one "missing" byte and this is the low order byte of that word. Analysis both ways of the first 2^{16} 65,536) key bytes generated gives the same chi-squared probability of 0.6599 i.e. non-significant at the 99.9% level. Analysis of the stepping of the individual bytes shows structure with, of course, the most significant bytes stepping least often.

One APT (Unknown, 2020, p. 7) for a stream length, again, of 4096, produced 32 values of key each with a frequency count of 128. This was program was run twice on two different starting values with the same out but slightly overlapping key values – all in the low 4,000 range.

5.14. Stuxnet Bot Configuration Data

This data (Matrosov, 2011, pp. 65, 74) is stored in 1860 bytes. The decryption algorithm is presented in the paper's Appendix B (although the paper says it is Appendix A). The Python code in the appendix was converted to C code and comments added. All integers were defined as int64_t to minimise buffer overflow.

It is unclear from the code the meaning of key, counter and sym. However, it may be inferred that: key comes from a set of values (one of the two 32-byte Hex streams perhaps?) to seed the decryption algorithm; counter is a positional counter on the 1860 long text; and sym is the encrypted value to be decrypted. Analysing the code in segments:

```
v0 = key * counter;
v1 = v0 >> 0xb;              /* Bitwise right shift      11 */
v1 = (v1 ^ v0) * 0x4e35;     /* Bitwise XOR multiplied by 20021 */
v2 = v1 & 0xffff;            /* Bitwise AND */
v3 = v2 * v2;
v4 = v3 >> 0xd;              /* Bitwise right shift 13 */
v5 = v3 >> 0x17;             /* Bitwise right shift 23 */
```

it is clear that: any combination of key and counter that is zero results in v0 being zero; and hence v1 being zero; any value of v0 less than 2^{11} - 1 (2047) will be result in intermediate v1 being zero (as only zeros will be right shifted); a v0 32-bit integer with the 16 least significant bits set to zero will result in v1 being zero (there are 2^{16} (65536) such numbers) any combination of key and counter that is zero results in v0 being zero; and hence v1 being zero. This paper notes that:

- any value of v0 less than 2^{11} - 1 (2047) will be result in intermediate v1 being zero (as only zeros will be right shifted);

- a v0 32-bit integer with the 16 least significant bits set to zero will result in v1 being zero (there are 2^{16} (65536) such numbers);
- any outcome which results in v3 being zero will result in v4 and v5 being zero.

This paper postulates that there might a lot of outcomes of zero so a program was written to exhaust all 256 values of key and counter. The decryption routine completed and returned the value of the line one before the original code.

From all 256*256 outcomes 256 zeros would be expected, 785 were seen (expected 256) and a chi-squared test on the list of frequency counts gives a probability of 0. The program was run for 256 possible key values and 1860 (counter) positions of encryption/decryption. 4296 zeros seen (expected 1860) and a similar chi-squared test gives a probability of 0. The program was run for both 31-byte hex key (Falliere, 2011, p. 22); 10 zeros were seen, 7.3 expected and a similar chi-squared test gives a probability of 0.21. Eight zeros were seen again 7.3 expected, and a similar chi-squared test gives a probability of 0.0086.

The encryption algorithm has several limitations. These are apparent by looking at the code. However, they are much less of an issue for a recurring key stream with well-chosen or, at least, not poorly chosen values.

5.15. Crypto-variable Generation

This algorithm (NTT-Security-Holdings, 2021, p. 19) shows how the malware generates a key from a seed and has severe limitations:

Initialise seed.

val1 = (seed&1)|(seed<<16)&0xFFFFFFFF;

val2 = (seed>>16)|(seed&0x00001000);

CV = ((val1<<8)&0xFFFFFFFF) | ((val2)>>8)&0xFFFFFFFF);

Analysing the first 2^{16} values: for val2, (seed&0x00001000) will be x00001000 if an only if bit 3 is 1. Therefore, it will be 0 half the time on average. A 16-bit right shift brings all zeros to the 16 least significant bits so the subsequent AND will always produce all zeros. It is clear that there is some structure to be tested.

The first 2^{16} (0-65,535) were generated and found to repeat after 512 seeds. Each crypto-variable is generated 128 times. For 2^{17} (0-131,071) the same 512 crypto-variables were generated. It is the same for 2^{18}. for 2^{19}, it is the same but the crypto variables are generated 1024 times. All are even numbers. For 2^{20} the spreadsheet was unable to read all of the values.

For further analysis Monte Carlo sampling was used with a Confidence Level of 99.9% and 1% margin of error. For a population size of 2^{32} a sample size of 27061 crypto-variables would be needed. These were generated producing a data set of 21133 unique crypto-variables each with a frequency of 1 to 6. 21133 is not an integer power of 2 and lies between 2^{14} (16,384) and 2^{15} (32,768). This is approximately$1/2^{17}$ (1/131,072) the size of the seed space. Given that the size of previous subsets of crypto-variables are a power of 2 the Monte Carlo results are a little disappointing. One of the numbers with a frequency count of 6 is 2,197,815,800 which is greater than 2^{31}. It is clear that that a seed space of 2^{32} reduces to a crypto-variable space much smaller, possibly 2^{16}.

The Monte Carlo simulation was extended to 2^{19} random seeds. This produced 64971 crypto-variables which is just less than 2^{16} (65,536). A second run of this program produced 64919 crypto-variables I.e., very similar to the first run. Combining both results and deduplicating produced 65521 unique crypto-variables, still less 2^{16} (65,536).

It is inferred from these Monte Carlo simulations that the crypto-variable space is 2^{16} and, therefore, within the realms of a Brute-force attack. As with the Stuxnet algorithm there are limitations to this algorithm and it des not align with Shannon's fourth point on minimising error propagation.

5.16. Customised Hashing

On APT (Soo, 2017) uses a customised hashing routine.

5.17. Standard Algorithms That Have Been Modified

A Tiny Encryption Algorithm (TEA) implementation has been seen that uses a key modified during encryption and decryption operations (Fidelis-Cybersecurity-Solutions, 2014, p. 43).

Custom implementation have been observed for example of AES-256-CBC (Marczak, 2014). RC6 key setup has been modified (Kaspersky, 2015d). Another RC4 routine was

poorly modified (Settle, 2016b) while yet another uses modified RC4 (Grunzweig, 2018). Another APT (D. Huss, 2017, p. 6) uses the Windows executable PowerSpritz. This hides payload and malicious PowerShell command using a non-standard implementation of Spritz, an RC4 variant. The modification of RC4 in at least two attacks may be accidental. Researchers note (Talos, 2018) that the malware authors incorrectly implemented initialisation of the S-boxes. Kaspersky imply implementation invocation of RC5 as specific to one APT (Kaspersky, 2013a, p. 27).

One APT (Airbus-D&S-CyberSecurity-blog, 2014) modifies Base64 to avoid "/" as it works on URLs which contain that character.

Another APT(DAHAN, 2017, p. 13) uses software that is based on DiskCryptor, a legitimate disk encryption utility.

5.18. Use of Victim's Machine Data

At least one APT (CrowdStrike, 2014c, p. 27) uses the infected machine's hard disk serial number, XOR'ed with the a eight-long key and nibble-wise encoded as upper-case ASCII characters in the range (A-P).

Another APT (van Dantzig, 2015, pp. 11-29) encodes strings from key which is made of dynamic values from the process stack. This will only work with static class and calling method names are static.

Key has been derived from system variables e.g., GetTickCount (Ferrer, 2010, p. 16). This custom algorithm encrypts again using a bitwise NOT. GetTickCount is also used to iniitialise srand (Checkpoint-Research, 2021, p. 12). Although not an encryption method, as such (there is no way to reverse it) the result is used to populate a variable called "new_user_id" which is then used in the routine writefile, presumably to pass to another stage of the attack.

In another (unspecified) encryption algorithm (Trend-Micro-Threat-Research-Team, 2012, p. 6) the key used for the is the machine's MAC address. Use of the MAC address is seen elsewhere (Villeneuve, 2012b, p. 5) where the values of the addressed are increased by 1 and the result used as an encryption key.

Similarly (Raiu, 2013, pp. 5-6) states that parts of a malicious DLL file are encrypted with system configuration related information. This ensures it will only work properly on the victim's system.

6. Miscellaneous

One APT (B. Levene, 2018) used HTML containing a vbscript to encode PowerShell commands which were then executed.

Use of Metasploit's Shikata Ga Nai encoder (Legezo, 2018, pp. 1-2, 4-6) is noted. A wider discussion of this is published by Mandiant (MILLER, 2019). While another APT uses Shikata Ga Nai for 32-bit shellcode and an the XOR dynamic encoder for 64-bit shellcode (Horejsi, 2020, p. 6).

Data has been seen to be "hexified" (Falliere, 2011, p. 22) to transform binary data into ASCII e.g.0x12, 0x34 "1234".

One APT in a multi-stage attack (GReAT, 2018b, pp. 4, 19) has an encryption algorithm that reminds the analysts of PKZIP encryption but seems to be modified. Also, probably Acid Cryptofiler military grade encryption software is observed. The same APT sends the data to the C&C server compressed with Zlib, encrypted with a modified PKZIP stream cipher and then Base64-encoded (GReAT, 2018c, p. 6).

FireEye (Siedlarz, 2016) highlight the evolution of one APT. The APT used Base64 and RC4 later changing the algorithm to include an intermediate step of an XOR plus a 12-byte salt.

7. Inability to Process Encryption Algorithms

Several algorithms could not be modelled, either because they appeared incomplete or they were hard to read as they had blurry images (Qihoo-360-Technology, 2018). Most assembler programs were not emulated (Avisa-Partners, 2020c, p. 34), (telecom.com, 2020).

One APT encrypted data with a algorithm lifted from a game server engine written by a group named "My Destiny Team." (FireEye, 2014a), No further information is provided.

At least two APTs used unknown schemes, unknownVB6 crypter (Scott-Railton, 2015, p. 13) or an encryption scheme that the analysts were unable to identify (Novetta, 2016a, p. 7).

One description (Diogos, 2017, p. 29) highlights a 10-long key with the XOR encoding “skipping every 3 bytes”. It is not clear what is meant by this and the authors state that they have not reproduced the decryption routine to maintain brevity.

One positional decryption description (Alexander, 2017, p. 12) claims to be a decryption algorithm but part of the algorithm looks like it uses the encrypted text to encrypt. However, for the purpose of this paper it was assumed that the decrypted character replaces its encrypted character in the buffer. This algorithm was programmed in C for all possible 8-bit ASCII values at the first 10 positions, multiple values decrypted/encrypted to the same value. Values started at a multiple of 32 and repeated after 32 steps. This discovery adds weight to the argument that the description provided may not be correct. Another description (cybergeeks, 2021) also suggests the need for the encrypted character to encrypt. Another algorithm (Avisa-Partners, 2020c, p. 34) appears to be a 1to Many mapping for decryption.]

One (ESET, 2018, p. 12) was not attempted due to complexity: an obfuscation algorithm that even the authors of the white paper state that “the code becomes far more complex to analyze for both malware researchers and automatic algorithms in security software.”.

8. Discussion

It is clear from the examples that bespoke encryption does necessarily mean better encryption. However, the use of these encryption schemes does not indicate lack of cryptographic skill and knowledge. This paper asserts that a good APT should use an appropriate level of encryption for the required attack and it is clear that some of these algorithms perform that function. This paper also asserts that attackers should display to defensive analysts a certain level of cryptographic knowledge. No more and no less. In addition, a good APT may wish to deceive by displaying a level lower of knowledge or skill than they, the APT, possess.

Alignment of attacks to Shannon’s Law is mixed. For example: Complexity of enciphering and deciphering is not minimised; Error propagation has been observed; Base64 etc encoding ensures that the encrypted message is longer than the plain text. However: the

amount of secrecy appears to be proportionate to the effort put in to securing the data; and key sizes seem to be as small as possible.

For analysts one of the lessons of this review is that cipher text should be aligned for analysis both from the start and the end. The former will pick up encryption that starts from the start and works forward, the later will pick up encryption that starts from the end and works backwards. All software used by analysts should be written to perform this function.

There is sufficient evidence to suggest that not all bespoke cryptographic algorithms mean stronger cryptography but the level of cryptography demonstrated by APTs may be enough for the level of secrecy and obfuscation desired.

9. A Critique of this Work and Suggested Further Lines of Work

The idea behind this paper was to review as many cryptographic methodologies as possible and highlight potential issues. The paper is restricted mainly to Windows high-level code and good prose descriptions. Some assembler analysis was performed but more complex assembler routines were placed out of scope. More work needs to be done on those, and non-Windows, routines as well as on the cryptographic variable production.

The search for encoding and decoding could have used the terms “encod” and “decod” instead of “encode” and “decode”. The estimate for the number of attacks using Base64 is just that - an estimate.

Not all of the algorithms could be exactly reproduced as not all white papers gave a complete description – some, for example, did not declare variables in the code presented and sensible choices of when to use a 32-bit or 64-bit word had to be made.

Adobe Advanced search seems to have a limit issue with respect to the number of results with – three searches (one related to other research) returned the figure of 500 and an error message. The “crypt” and “obfusct” searches were then performed using a Windows Microsoft search.

There may exist white papers and hence, encryption algorithms not analysed in this paper.

It is acknowledged that some observations may be the fault of this author, both in interpretation of the data or C coding or analysis.

10. Concluding Remarks

The paper has presented and discussed encryption, compression and obfuscation techniques used by APTs. It has shown that, apart from internationally acceptable techniques, bespoke algorithms are also used. All of the techniques and algorithms, generally, serve the purpose. Many algorithms adhere to Shannon's Law but some do not. However, without evidence of the underlying business philosophy of the different APTs, it is difficult to for a view of the rational for a few of the algorithms.

REFERENCES

Airbus-D&S-CyberSecurity-blog. (2014) 'Leouncia and Orcarat' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Alexander, G. D., Jakub; Crete-Nishihata, Masashi; Brooks, Matt (2017) *Insider Information: An Intrusion Campaign Targeting Chinese Language News Sites.*

Alintanahin, K. (2015) *Operation Tropic Trooper Relying on Tried-and-Tested Flaws to Infiltrate Secret Keepers.*

Alperovitch, D. (2014) *Deep in Thought: Chinese Targeting of National Security Think Tanks.*

amnesty.org (2020) *German-Made Finspy Spyware Found in Egypt, and Mac and Linux Versions Revealed.*

ANSSI. (2020) 'Development of the Activity of the Ta505 Cybercriminal Group ' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Antiy-CERT (2015) *Analysis on Apt-to-Be Attack That Focusing on China's Government Agency*.

Avisa-Partners (2020a) *The Lazarus Constellation.*

Avisa-Partners (2020b) *The Lazarus Constellation.*

Avisa-Partners (2020c) *The Lazarus Constellation.*

BAe-Systems (2014a) *Snake Campaign & Cyber Espionage Toolkit.*

BAe-Systems (2014b) *Snake Campaign & Cyber Espionage Toolkit.*

Bar, T. (2017) 'Targeted Attacks in the Middle East Using Kasperagent and Micropsia' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Bar, T. C., Simon (2016) 'Prince of Persia: Infy Malware Active in Decade of Targeted Attacks' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Benchea, R. M., Alexandru; Vatamanu, Cristina;; Luncasu, Victor (2015) *Apt28 under the Scope a Journey into Exfiltrating Intelligence and Government Information.*

Bitdefender (2016) *Pacifier Apt*

BLACKBERRY-RESEARCH-&-INTELLIGENCE-TEAM (2020) *Bahamut: Hack-for-Hire Masters of Phishing, Fake News, and Fake Apps.*

BlackBerry (2020) *Decade of the Rats. Cross-Platform Apt Espionage Attacks Targeting Linux, Windows and Android.*

Breitenbacher, D. O., Kaspars (2020) *Operation in(Ter)Ception: Targeted Attacks against European Aerospace and Military Companies.*

Brumaghin, E. a. o. C. T. r. (2019) 'Sweed: Exposing Years of Agent Tesla Campaigns' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st August 2023).

Chang, Z. L., Kenney ; Luo, Aaron ; Pernet, Cedric; Yaneza, Jay (2015) *Operation Iron Tiger: Exploring Chinese Cyber-Espionage Attacks on United States Defense Contractors.*

Check-Point (2015) *Volatile Cedar.*
Checkpoint-Research (2021) *Indigozebra Apt Continues to Attack Central Asia with Evolving Tools.*
Chen, J. C. L., Kenney ; Horejsi, Jaromir; Chen, Gloria (2021) 'Biopass Rat: New Malware Sniffs Victims Via Live Streaming' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
CHEREPANOV, A. (2016) *Operation Groundbait: Analysis of a Surveillance Toolkit.* eset
Cisco-Talos-Intelligence-Group. (2021) 'Sowing Discord: Reaping the Benefits of Collaborationapp Abuse ' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
ClearSky-Cyber-Security-ltd (2021) *"Lebanese Cedar" Apt Global Lebanese Espionage Campaign Leveraging Web Servers.*
Co-Authored-by-Rapid7, I.-G. (2019) 'Apt10 Targeted Norwegian Msp and Us Companies in Sustained Campaign' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Command-Five-Pty-Ltd (2011) *Sk Hack by an Advanced Persistent Threat.*
Command-Five-Pty-Ltd (2012a) *Command and Control in the Fifth Domain.*
Command-Five-Pty-Ltd (2012b) *Command and Control in the Fifth Domain.*
Cox, A. E., Chris ; Gragido, Will ; Harrington, Chrisl Jon , McNeill (2012) *The Voho Campaign: An in Depth Analysis.*
Crowdstrike-Global-Intelligence-Team (2015) *Deep Panda.*
CrowdStrike (2014a) *Putter Panda.*
CrowdStrike (2014b) *Putter Panda.*
CrowdStrike (2014c) *Putter Panda.*
Cutler, S. (2012) *The Mirage Campaign.*
Cyber-Geeks. (2021) 'A Step-by-Step Analysis of the New Malware Used Byapt28/Sofacy Called Skinnyboy' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Cyber-Safety-Solutions-Team. (2019) 'New Slub Backdoor Uses Github, Communicates Via Slack' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
cybergeeks (2021) *A Detailed Analysis of Elmer Backdoor Used by Apt16.*
CyCraft-Research-Team (2020) *Apt Group Chimera - Apt Operation Skeleton Ket Targets Taiwan Semiconductor Vendors.*
Cylance (2018) *The Spyrats of Oceanlotus*
DAHAN, A. (2017) *Night of the Devil: Ransomware or Wiper? A Look into Targeted Attacks in Japan Using Mbr-Oni.*
DATA-SecurityLabs, G. (2014) *Operation "Toohash" How Targeted Attacks Work.*
Dela Paz, R. (2012) *The Heartbeat Apt Campaign.*
Denbow, S. a. H., Jesse (2012) *Pest Control: Taming the Rats.*
Diogos, T. D., Sachin ; Mendrez, Rodel (2017) *Operation Grand Mars: Defending against Carbanak Cyber Attacks.* Trustwave Spider Labs

Doctor-Web (2020a) *Study of the Shadowpad Apt Backdoor and Its Relation to Plugx.*
Doctor-Web (2020b) *Study of the Shadowpad Apt Backdoor and Its Relation to Plugx.*
Dr.Web (2020a) *Study of the Apt Attacks on State Institutions in Kazakhstan and Kyrgyzstan.*
Dr.Web (2020b) *Study of the Apt Attacks on State Institutions in Kazakhstan and Kyrgyzstan.*
Dr.Web (2020c) *Study of the Apt Attacks on State Institutions in Kazakhstan and Kyrgyzstan.*
Draco-Team (2020) *Dissecting a Chinese Apt Targeting South Eastern Asian Government Institutions.*
Ehrrlich, A. B. S. (2021) *From Wiper to Ransomware: The Evolution of Agrius.*
Elastic. (2020) 'A Close Look at the Advanced Techniques Used in a Malaysian Focused Apt Campaign' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
ESET-Research-Whitepapers (2018) *Turla Outlook Backdoor Analysis of an Unusual Turla Backdoor.*
eset (2016) *En Route with Sednit Part 2.*
eset (2019a) *Greyenergy: A Successor to Blackenergy.*
eset (2019b) *Greyenergy: A Successor to Blackenergy.*
eset (2019c) *Machete Just Got Sharper.*
eset (2019d) *Machete Just Got Sharper.*
eset (2020) *Invisimole: The Hidden Part of the Story Unearthing Invisimole's Espionage Toolset and Strategic Cooperations.*
ESET, s. s. r. o. (2018) *Diplomats in Eastern Europe Bitten by a Turla Mosquito.*
Expert; Park, S. (2020) *Lazarus Covets Covid-19-Related Intelligence.*
Fagerland, S. G., Waylon (2014) *The Inception Framework: Cloud-Hosted Apt.*
Falcone, R. (2017) 'Second Wave of Shamoon 2 Attacks Identified' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Falcone, R. (2018) 'New Threat Actor Group Darkhydrus Targets Middle East Government' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
FALCONE, R., GRUNZWEIG,, JOSH; MILLER-OSBORN, JEN; OLSON, RYAN (2015) *Operation Lotusblossom.*
Falcone, R. M.-O., Jen (2016) 'Emissary Trojan Changelog: Did Lotus Blossom Cause It to Evolve' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Falliere, N. O. M., Liam; Chien, Eric (2011) *W32.Stuxnet Dossier.*
Faou, M. (2020) *From Agent.Btz to Comrat V4 a Ten-Year Journey.*
Faou, M. (2020) *From Agent.Btz to Comrat V4 a Ten-Year Journey.* eset.
Faou, M. T., Mathieu ;TDupuy, homas (2019a) *Operation Ghost the Dukes Aren't Back — They Never Left.*
Faou, M. T., Mathieu ;TDupuy, homas (2019b) *Operation Ghost the Dukes Aren't Back — They Never Left.*
Ferrer, Z. C. F., Methusela (2010) *In-Depth Analysis of Hydraq the Face of Cyberwar Enemies Unfolds.*
Fidelis-Cybersecurity-Solutions (2014) *Intruder File Report- Sneakernet Trojan.*

Fidelis-Cybersecurity-Solutions, G.-D. (2013) *Fidelis Threat Advisory #1009 "Njrat" Uncovered.*
Fidelis-Cybersecurity-Solutions, G.-D. (2015) *Dissecting the Malware Involved in the Inocnation Campaign.*
Fidelis (2014) *New Cdto: A Sneakernet Trojan Solution.*
FireEye. (2014a) 'Forced to Adapt: Xslcmd Backdoor Now on Os X' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
FireEye (2014b) *Poison Ivy: Assessing Damage and Extracting Intelligence.*
FireEye. (2014c) 'Trojan.Apt.Seinup Hitting Asean' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January2023).
FireEye (2015) *Apt30 and the Mechanics of a Long-Running Cyber Espionage.*
FireEye (2019a) *Double Dragon Apt41, a Dual Espionage and Cyber Crime Operation.*
FireEye (2019b) *Double Dragon Apt41, a Dual Espionage and Cyber Crime Operation.*
FireEye. (2019c) 'Mahalo Fin7: Responding to the Criminal Operators' New Tools and Techniques' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Gandotra, E. B., Divya ; Sofat, Sanjeev (2014) 'Malware Analysis and Classification: A Survey', *Journal of Information Security,* 5, pp. 56-64.
Global-Research-and-Analysis-Team (2013) *The Nettraveler (Aka 'Travnet').*
Global-Research-and-Analysis-Team (2016) *The Projectsauron Apt. Technical Analysis 2-16.*
Gorelik, M. G., Alon; Braga, Bruno (2019) *New Campaign Delivers Orcus Rat.* Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections
GovCERT.ch (2016) *Apt Case Ruag Technical Report.*
GReAT. (2016a) 'Cve-2015-2545: Overview of Current Threats' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
GReAT (2016b) *Projectsauron: Top Level Cyber-Espionage Platform Covertly Extracts Encrypted Government Comms.* Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections
GReAT (2018a) *"Red October". Detailed Malware Description 1. First Stage of Attack.*
GReAT (2018b) *"Red October". Detailed Malware Description 3. First Stage of Attack.*
GReAT (2018c) *"Red October". Detailed Malware Description 4. First Stage of Attack.*
GReAT (2018d) *"Red October". Detailed Malware Description 5. Second Stage of Attack.*
GReAT (2019) *Gaza Cybergang Group1, Operation Sneakypastes.*
GReAT, E. (2021) 'Apt10: Sophisticated Multi-Layered Loader Ecipekacdiscovered in A41apt Campaign' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Gross, J. (2015) *Cylance Spear Team: A Threat Actor Resurfaces.*
Gross, J. e. a. (2016) *Operation Dust Storm.*
Group-IB (2018) *Silence Moving into the Darkside.*
Grunzweig, J. (2018) 'The Tophat Campaign: Attacks within the Middle

East Region Using Popular Third-Party Services' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Guarnieri , C. S., Mark (2014) *Keyboy, Targeted Attacks against Vietnam and India.*
Haq, T. M., Ned ; Vashisht, Sai; Scott, Mike (2014a) *Operation Quantum Entanglement.*
Haq, T. M., Ned ; Vashisht, Sai; Scott, Mike (2014b) *Operation Quantum Entanglement.*
Hegel, T. (2018) *Burning Umbrella, an Intelligence Report on the Winnti Umbrella and Associated State-Sponsored Attackers.*
Helios-Team (2016a) *Operation Oniondog.*
Helios-Team (2016b) *摩诃草组织.*
Hiroaki, H. L., Ted (2021) *Earth Baku an Apt Group Targeting Indo-Pacific Countries with New Stealth Loaders and Backdoor.*
Horejsi, J. (2018) 'New Powershell-Based Backdoor Found in Turkey, Strikingly Similar to Muddywater Tools' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2030).
Horejsi, J. C., Joseph C. (2020) *Operation Overtrap Targets Japanese Online Banking Users Via Bottle Exploit Kit and Brand-New Cinobi Banking Trojan.*
Hromcová, Z. (2019) 'Eset Discovers Attor, a Spy Platform with Curious Gsm Fingerprinting' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Hromcová, Z. C., Anton (2020) *Invisimole: The Hidden Part of the Story Unearthing Invisimole's Espionage Toolset and Strategic Cooperations.*
Huss, D. (2017) *North Korea Bitten by Bitcoin Bug.*
Huss, D. L., Selena (2021) *Triple Threat: North Korea-Aligned Ta406 Steals, Scams and Spies.*
Huss, D. M., Matthew (2017) *Operation Rat Cook: Chinese Apt Actors Use Fake Game of Thrones Leaks as Lures |*
Hwa, C. R. (2014) 'Trojan.Apt.Banechant: In-Memory Trojan That Observes for Multiple Mouse Clicks' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Insikt-Group (2020) *Chinese State-Sponsored Group 'Reddelta' Targets the Vatican and Catholic Organizations.*
Jazi, H. (2021) *Lazyscripter: From Empire to Double Rat.*
JPCERT-CC. (2016) 'Asruex: Malware Infecting through Shortcut Files' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Kaspersky (2013a) *The Regin Platform Nation-State Ownership of Gsm Networks.*
Kaspersky (2013b) *The Regin Platform Nation-State Ownership of Gsm Networks.*
Kaspersky (2014a) *Crouching Yeti — Appendixes.*
Kaspersky (2014b) *Unveiling "Careto" - the Masked Apt.*
Kaspersky (2015a) *The Duqu 2.0 Technical Details Version: 2.1.* Kaspersky.
Kaspersky (2015b) *The Duqu 2.0 Technical Details Version: 2.1 (11 June 2015).*
Kaspersky (2015c) *Equation Group: Questions and Answers.*
Kaspersky (2015d) *Equation Group: Questions and Answers.*

Kaspersky (2015e) *Wild Neutron – Economic Espionage Threat Actor Returns with New Tricks,.*
Klijnsma, Y. H., Danny ; Sahertian,, Mitchel ; de Mik, Krijn ; van Dantzig, Maarten ; Yun Zheng Hu, Haagsma, Lennart ; van Hensbergen, Martin de Jong, Erik (2016) *Mofang. A Politically Motivated Information Stealing Adversary.*
Knuth, D. E. (1981) *The Art of Computer Programming. 2nd Edn.* Vol. 2. Reading, Mass: Addison-Wesley (Addison-Wesley series in computer science and information processing).
Lee, B. G., Josh. (2018) 'Sofacy Attacks Multiple Government Entities' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Lee, B. G., Josh (2015) 'Bbsrat Attacks Targeting Russian Organuzations Linked to Roaming Tiger' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Lee, B. G., Josh (2015) *Watering Hole Attack on Aerospace Firm Exploits Cve-2015-5122 to Install Isspace Backdoor.*
Legezo, D. (2018) *Luckymouse Hits National Data Center to Organize Country-Level Waterholing Campaign.*
Levene, B. (2018) 'Sure, I'll Take That! New Combojack Malware Alters Clipboards to Steal Cryptocurrency' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Levene, B. F., Robert; Miller-Osborn, Jen (2015) *Musical Chairs: Multi-Year Campaign Involving New Variant of Gh0st Malware.*
Levene, B. G., Josh; Ash, Brittany (2018) 'Patchwork Continues to Deliver Badnews to the Indian Subcontinent' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Lifshitz, N. E., Asaf Eitani, Amitai Ben Shushan ; Kushnir,Amnon ; Biton, Gil ; Korman, Martin; Shohat, Itay; Zilberstein, Arie (2021) *Tg1021: "Praying Mantis" Dissecting an Advanced Memory-Resident Attack.*
Lunghi, D. P., Cedric ; Lu, Kenney ; Yaneza, Jamz (2020) *Uncovering Drbcontrol Daniel Lunghi, Cedric Pernet, Kenney Lu, and Jamz Yaneza inside the Cyberespionage Campaign Targeting Gambling Operations.*
M-TRENDS (2021) *Special Report M-Trends.*
M.Léveillé, M.-E. T., Mathieu (2019) *Connecting the Dots Exposing the Arsenal and Methods of the Winnti Group.*
MalwarebytesLABS (2018) *Cybercrime Tactics and Techniques: Q2 2018.*
Mandiant (2013) *Apt1 Exposing One of China's Cyber Espionage Units.*
Mandiant, F. (2020) *M-Trends 2020.* MANDIANT, F. Available at: https://mandiant.widen.net/s/5pwlhgqt5t/m-trends-2020-report (Accessed: 29th April 2022).
Marczak, W. R. S.-R., John; Marquis-Boire, Morgan ; Paxson, Vern (2014) *23rd USENIX Security Symposium.* San Diego, CA.
Marschalek, M. (2014) *Evilbunny Suspect #4.*

Matrosov, A. R., Eugene; Harley, David; Malcho, Juraj (2011) *Stuxnet under the Microscope Revision 1.31.*

McAfee-Labs. (2020) 'Operation (노스 스타) North Star a Job Offer That's Too Good to Be True?' Available at: (Accessed: 31st January 2023).

Mercer, W. R., Paul; Ventura, Vitor. (2020) 'Promethium Extends Global Reach with Strongpity3 Apt' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Microsoft-Security. (2021) 'Breaking Down Nobelium's Latest Early-Stage Toolset' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Microsoft. (2021) 'Foggyweb: Targeted Nobelium Malware Leads to Persistentbackdoor' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

MILLER, S. R., EVAN ; CARR, NICK (2019) 'Shikata Ga Nai Encoder Still Going Strong' *THREAT RESEARCH*, Available at: https://www.mandiant.com/resources/blog/shikata-ga-nai-encoder-still-going-strong (Accessed: 9th May 2023).

Moran, N. a. V., Nart (2014) 'Survival of the Fittest: New York Times Attackers Evolve Quickly' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

netlab.360.com. (2020) 'Dacls, the Dual Platform Rat' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Novetta (2014a) *Derusbi (Server Variant) Analysis*

Novetta (2014b) *Derusbi (Server Variant) Analysis*

Novetta (2014c) *Hikit Analysis.*

Novetta (2016a) *Operation Blockbuster Destructive Malware Report.*

Novetta (2016b) *Operation Blockbuster Destructive Malware Report.*

NTT-Security-Holdings (2021) *Report on Apt Attacks by Blacktech.*

Palo-Alto-Networks-Blog. (2016) 'New Webkey Attacks Use Dns Requests as Command and Control Mechanism' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Palumbo, P. (2014) *W64/Regin, Stage #1.*

Paul, R. (2013) *Apt1: Technical Backstag.*

Perigaud, F. (2014a) *Plugx: Some Uncovered Points.*

Perigaud, F. (2014b) *Plugx: Some Uncovered Points.*

Perigaud, F. B., Boris. (2014) 'Vinself Now with Steganography' *Airbus D&S CyberSecurity blog*, Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Pernet, C. S., Eyal (2015) *The Spy Kittens Are Back: Rocket Kitten 2.*

Pernet, C. S., Eyal (2015) *The Spy Kittens Are Back: Rocket Kitten 2.*

profero (2021) *Apt27 Turns to Ransomware.*

proofpoint (2015) *The Shadow Knows: Malvertising Campaigns Use Domain Shadowing to Pull in Angler Ek.*
ptsecurity. (2021) 'Lazarus Group Recruitment: Threat Hunters Vs Head Hunters' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Qihoo-360-Technology. (2018) 'Analysis of Cve-2018-8174 Vbscript 0day and Apt Actor Related to Office Targeted Attack,' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
QuoIntelligence (2020) *Winnti Group: Insights from the Past.*
Raiu, C. B., Kurt (2014) *Nettraveler Apt Gets a Makeover for 10th Birthday.*
Raiu, C. S., Igor ; Baumgartner, Kurt ; Kamluk, Vitaly (2013) *The Miniduke Mystery: Pdf 0-Day Government Spy Assembler 0x29a Micro Backdoor (or 'How Many Cool Words Can You Fit into One Title').*
Rostovcev, N. (2020) *The Footprints of Raccoon: A Story About Operators of Js-Sniffer Fakesecurity Distributing Raccoon Stealer.*
RSA-RESEARCH (2017) *Kingslayer– a Supply Chain Attack.*
RSA (2014a) *Shell_Crew.*
RSA (2014b) *Shell_Crew.*
RSA (2014c) *Shell_Crew.*
Schneier, B. (2007) *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* john Wiley & Sons.
Scott-Railton, J. M.-B., Morgan ; Guarnieri, Claudio Marschalek, Marion (2015) *Packrat: Seven Years of a South American Threat Actor.*
Scott, J. S., Drew (2016) *Know Your Enemies*
securelist. (2015) 'Inside the Equationdrug Espionage Platform' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
securityintelligence.com. (2019) 'More_Eggs, Anyone? Threat Actor Itg08 Strikes Again' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Serper, A. (2020) 'Who's Hacking the Hackers: No Honor among Thieves' *maliciouslife*, Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Settle, A. D., Bapadittya; Griffin, Nicholas; Toro, Abel (2016a) *Jaku. Analysis of a Botnet Campaign.*
Settle, A. D., Bapadittya; Griffin, Nicholas; Toro, Abel (2016b) *Jaku. Analysis of a Botnet Campaign.*
Shannon, C. E. (1949) 'Communication Theory of Secrecy Systems', *The Bell System Technical Journal 28(4)*, pp. 656 - 715.
Shevchenko, S. N., Adrian (2017) 'Lazarus' False Flag Malware' *BAE SYSTEMS THREAT RESEARCH BLOG*, Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Siedlarz, M. D., Kristen. (2016) *RSA Conference 2016*. Singapore.

Singh, G. S. (2013) 'A Study of Encryption Algorithms (Rsa, Des, 3des and Aes) for Information Security', *International Journal of Computer Applications (0975 – 8887),* 67(April 2013), pp. 33-38.
Singh, S. (2020) 'Attack on Indian Government, Financial Institutions' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Singh, S. Y. H. C. (2016) 'Targeted Attacks against Banks in the Middle East' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
sKyWIper-Analysis-Team (2012a) *Skywiper (A.K.A. Flame A.K.A. Flamer): A Complex Malware for Targeted Attacks.* Lab), L. o. C. a. S. S. C.
sKyWIper-Analysis-Team (2012b) *Skywiper (A.K.A. Flame A.K.A. Flamer): A Complex Malware for Targeted Attacks.* Lab), L. o. C. a. S. S. C.
Sofacy (2015) *Sofacy Apt Hit High Profile Targets.*
Soo, J. G., Josh (2017) 'Recent Inpage Exploits Lead to Multiple Malware Families' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st August 2023).
Spohn, M. G. (2012) *Know Your Digital Enemy Anatomy of a Gh0st Rat.*
Symantec-Security-Response (2014) *Regin: Top-Tier Espionage Tool Enables Stealthy Surveillance.*
Symantec (2014) *Dragonfly: Cyberespionage Attacks against Energy Suppliers 2014, P Xx.*
Symantec (2015a) *The Waterbug Attack Group.*
Symantec (2015b) *The Waterbug Attack Group.*
Symnatec (2015) *Butterfly: Corporate Spies out for Financial Gain.*
Talos. (2018) 'New Vpnfilter Malware Targets at Least 500k Networking Devices Worldwide' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st August 2023).
telecom.com. (2020) 'Lolsnif – Tracking Another Ursnif-Based Targeted Campaign' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
telsy. (2019) 'The Lazarus' Gaze to the World: What Is Behind the First Stone?' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Telsy (2020) *When a False Flag Doesn't Work: Exploring the Digital-Crime Underground at Campaign Preparation Stage.*
The-Cylance-Team (2016) *Nigerian Cybercriminals Target High-Impact Industries in India Via Pony.*
The-SecDev-Group (2009) *Tracking Ghostnet: Investigating a Cyber Espionage Network.*
ThreatConnect-Inc.-and-Defense-Group-Inc. (2015) *Camerashy Closing the Aperture on China's Unit 78020.*
ti.qianxin.com. (2019) 'Ttps://Ti.Qianxin.Com/Blog/Articles/Oceanlotus-Attacks-to-Indochinese-Peninsula-Evolution-of-Targets-Techniques-and-Procedure/' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).

Tikhonova, A. K., Dmitry. (2021) 'The Art of Cyberwarfare' *blog.group-ib.com*, Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
TIVADAR, M. B., Bíró ; ISTRATE, Cristian (2013a) *Closer Look at Miniduke*
TIVADAR, M. B., Bíró ; ISTRATE, Cristian (2013b) *Closer Look at Miniduke*
Tomonaga, S. (2015) *Analysis of a Recent Plugx Variant - "P2p Plugx".*
Tomonaga, S. N., Yuu (2015) *Revealing the Attack Operations Targeting Japan.*
Travers, T. (2017) *From Shamoon to Stonedrill Wipers Attacking Saudi Organizations and Beyond.*
Trend-Micro-Threat-Research-Team (2012) *The Taidoor Campaign: An in-Depth Analysis.*
Trend-Micro (2012) *2q Report on Targeted Attack Campaigns.*
Trend-Micro. (2018) 'Lazarus Continues Heists, Mounts Attacks on Financial Organizations in Latin America' *blog.trendmisro.com*, Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Trend-Micro. (2019) 'Ta505 at It Again: Variety Is the Spice of Servhelper and Flawedammyy' *TrendLabs Security Intelligence Blog*, Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Trend-Micro, C.-C.-S. (2017) *Operation Wilted Tulip Exposing a Cyber Espionage Apparatus.*
Unit-42. (2017) 'Threat Actors Target Government of Belarus

Using Cmstar Trojan' Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Unknown (2010) *Shadows in the Cloud: Investigating Cyber Espionage 2.0.*
Unknown (2020) *Grandoreiro: How Engorged Can an Exe Get?*
Unknown. (2022) *Cybermonitor/Apt_Cybercriminal_Campagin_Collections*. Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections]
us-cert-cisa (2020) *Mar-10292089-1.V2 – Chinese Remote Access Trojan:Taidoor.* Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections
van Dantzig, M. H., Danny; Ruiz, Frank; Klijnsma, Yonathan; Yun Zheng Hu; de Jong Erik; de Mik, Krijn; Haagsma, Lennart (2015) *Ponmocup. A Giant Hiding in the Shadows.*
Villeneuve, N. a. B., J. (2012a) *Detecting Apt Activity with Network Traffic Analysis. .*
Villeneuve, N. a. B., J. (2012b) *Detecting Apt Activity with Network Traffic Analysis. Trend Micro.*
Villeneuve, N. B., James T. ; Moran, Ned; Haq, Thoufique; Scott, Mike; Geers, Kenneth (2013) *Operation "Ke3chang": Targeted Attacks against Ministries of Foreign Affairs.*
Villeneuve, N. d. T., Jessa (2013) *Fakem Rat Malware Disguised as Windows®Messenger and Ahoo!® Messenger.*
Villeneuve, N. W., Kyle (2013) *Safe a Targeted Threat.*
Villeneuve, N. W., Kyle (2013) *Safe a Targeted Threat.*
welivesecurity (2020a) *Guildma: The Devil Drives Electric.*
welivesecurity (2020b) *Winnti Group Targeting Universities in Hong Kong.*
welivesecurity. (2021) 'Strategic Web Compromises in the Middle East with a Pinch of Candiru' Available at:

https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections (Accessed: 31st January 2023).
Wueest, C. A., Himanshu (2017) *Living Off the Land and Fileless Attack Techniques.*
Wyke, J. (2011) *What Is Zeus?* (Accessed: 28th January 20).
You, I. Y., K. (2010) *Proceedings of International conference on Broadband, Wireless Computing, Communication and Applications*. Fukuoka.
ZLAB (2018) *A Long-Term Espionage Campaign in Syria.*
zscaler (2011) *Alleged Apt Intrusion Set: "1.Php" Group.*