Computer Security Symposium 2018
22 - 25 October 2018

Cutwail 通信プロトコルの解析

阿曽村一郎[1]　武田康博[1]

概要：オンラインバンキングなどの認証情報を盗むマルウェアはバンキングトロジャンと呼ばれる．多くの場合，これらはメール本文 URL からのダウンロードさせるかたち，またはメールの添付ファイルとして配布される．バンキングトロジャンを配布するための基盤として，Cutwail というボットネットがある．Cutwail についてのこれまでの研究によれば，ボットと C & C サーバ間の通信は攻撃者が望むコンテンツのスパムメールを送信するためのメカニズムを持つことが分かっている．本稿では，Cutwail のボットと C & C サーバの間の通信に着目し，我々が観察した結果と以前の研究結果との比較結果を説明する．

キーワード：スパムメール, マルウェア, バンキングトロジャン, Cutwail, Dreambot

Communication protocol analysis of the Cutwail Spam Delivery Service

Ichiro Asomura[1]　Yasuhiro Takeda[1]

Abstract: A malicious computer program designed to gain access to confidential information stored or processed through online banking systems is called Banking Trojan. Most of Banking Trojan is distributed by a phishing emails sent from an infected computer that the criminal can remotely manage using a command and control server, the botnet. The bots connect directly to the command and control server, and receive instructions about the emails they should send. This report presents a detailed analysis on the communication protocol which is not reported in the previous studies.

Keywords: Spam, malware, Banking trojan, Cutwail, Dreambot

1. はじめに

本研究はバンキングトロジャンによる金融犯罪の被害軽減を目的としている．バンキングトロジャンによる被害軽減を実現するためにはバンキングトロジャンそのもの，つまりマルウェアに対する対策も必要だが，一方でバンキングトロジャンの配布を抑制することができれば被害軽減を実現する対策のひとつになりうる．不正送金を目的としたオンラインバンキングの認証情報窃取や，クレジットカードの不正使用を目的としたクレジットカード情報の窃取をするバンキングトロジャンの多くはスパムメール経由で配布されるため，我々はスパムメールの配信インフラに着目した．バンキングトロジャンの配布に使われるスパムメール配信インフラはいくつか確認されているが，その中のひとつに Cutwail ボットネットがある．Cutwail ボットネットは，2007 年頃にその存在が報告されており，これまでに 200 万台のコンピュータに感染し，1 日当たり 740 億通のスパムメッセージを送信していると考えられている [1]．Cutwail ボットネットは 2018 年現在でも犯罪者にスパムメール配信インフラとして活用されており，実に 10 年以上の歴史を持つ．Cutwail ボットネットの振る舞いに関わる解析結果は過去の研究でも報告されているが，Cutwail と C&C サーバ間の通信パケットを採取し，暗号化された通信内容を復号した上で通信内容を解析する手法は用いられていない．本稿では最新の Cutwail のボットと C&C サーバとの間の通信を解析し，過去の研究で報告されている内容と比較した結果，判明した新たな事実を説明する．将来的にバンキングトロジャンによる被害の軽減につなげていきたい．

[1] 株式会社みずほフィナンシャルグループ
Mizuho Financial Group

2. 関連研究

Cutwail についてはすでにいくつかの研究結果が報告されているが，以下に示す 3 つの研究については，Cutwail の振る舞いやデータ構造などが明らかにされている．[1], [2] では，マルウェア (バイナリ) 視点の解析結果と考察が報告されており，[3] では，C&C サーバ (ソースコード) 側の視点からの解析および検証の報告がされているため，本研究における主な参考文献とした．

- A study of the Pushdo / Cutwail Botnet [1]
- Pushdo's new second generation [2]
- Analysis of the Cutwail Botnet Command and Control Software [3]

2.1 A study of the Pushdo / Cutwail Botnet

2009 年に発表されたものであり，3 つの中で最初に Cutwail について報告されたものである．Cutwail だけでなく，Cutwail をインストールするマルウェアである Pushdo の振る舞いについても報告されている．ディスクへのアクセスを極力抑えることによってアンチウィルスソフトに検知されづらくする手法や，OS にデバイスドライバを組み込むことによって通信を監視し，メールアドレスなどの情報を入手することなどがあげられる．また，ボットと C&C サーバ間の暗号化された通信の復号方法から，通信データのデータ構造，通信の順序，およびボットとしての動作のフローチャートなどが詳細に説明されている．主にバイナリ解析と感染端末の解析からマルウェアの振る舞いを分析している．

2.2 Pushdo's new second generation

2013 年に発表された報告である．[1] と同じく，Pushdo と Cutwail の両方について報告している．通信データのデータ構造についても説明されており，その内容は [1] から進化した第二世代になっていることが報告されている．また，暗号化された通信を復号するための鍵が [1] で報告されたものとは異なっている．

2.3 Analysis of the Cutwail Botnet Command and Control Software

2015 年に発表された C&C サーバについての分析結果である．C&C サーバのソースコードを解析し，その仕様や振る舞いを明らかにした．また，C&C サーバの脆弱性を発見し，それを利用して C&C サーバに対する DDoS 攻撃を試みた結果の報告である．C&C サーバの脆弱性を明らかにする過程で通信データについても解析を行っている．通信データのデータ構造は [1] と同じであることが確認できる．

3. 観測環境

マルウェアの動作を観測するためには，実際にマルウェアを動作させる必要がある．しかし，何も対策せずに Cutwail を動かした場合，C&C サーバからの指示によってスパムメールを送信してしまう．そのためこの解析では，外部にスパムメールを送信させずにボットと C&C サーバとの通信の観測を行うための専用の観測環境を作成した．

Cutwail の C&C サーバの IP アドレスと C&C サーバとの通信は，通信プロトコルは SMTP では無いものの TCP のポート 25 または 443 を使うことが分かっており [2]，この通信は通す必要がある．しかし，実際にスパムメールを送るための通信は，インターネット上に存在する送信認証が設定されていない SMTP サーバを利用していると考えられるため，全ての IP アドレスに対するポート 25 と 443 の通信を許可するわけにはいかない．このため，NAT の機能を用いて C&C サーバ以外の全ての IP アドレスとの通信を，観測環境の内部に用意した疑似メールサーバの IP アドレスに書き換え，C&C サーバへのポート 25 に向けたパケットのみ外部へ通信可能とした．C&C サーバへの通信については，ISP 内でマルウェアの振る舞いとして検知およびブロックされることを防ぐ事，C&C サーバから直接観測環境の IP アドレスを見えないようにする事の二点から，観測環境の外部に借りた VPS のホストへ SSH ポート転送を行い，C&C サーバからは VPS ホストの IP アドレスが見える状態とした．

図 1 に観測環境の模式図を示す．

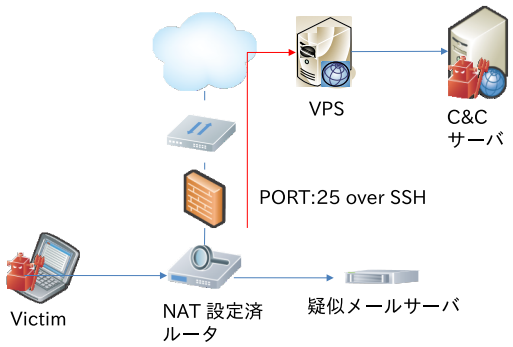

図 1: 観測環境

観測環境は一台の PC 上に仮想マシンを稼働させる形で作成した．仮想マシンホストである PC を，図 1 における NAT 設定済ルータとしても利用し，Victim と疑似メールサーバはこのホスト上でそれぞれ仮想マシンとして起動させた．Victim 自身で Cutwail からの通信先を変更すると，Victim で動作するボットプログラムに検知され振る舞いを変えてしまうことも考えられるため，通信先についての全ての制御は NAT 設定済ルータでのみ行う事と

した．SSH ポート転送は，疑似メールサーバから VPS に向けて行う事とした．したがって，NAT 設定済ルータでは，C&C サーバの IP アドレスに向けたポート 25 宛の通信を VPS へポート転送しているポートに向け，それ以外の全てのポート 25 宛の通信が疑似メールサーバのポート 25 を向くよう設定した．

表 1: 観測環境の構成

	仮想マシンホスト (設定済ルータ)	Victim	疑似メールサーバ
メーカー	HP	-	-
型番	EliteDesk 800 G2 SF	(仮想マシン)	(仮想マシン)
OS	Ubuntu 16.04	Windows7 32bit	Ubuntu 18.04

通信環境としては，観測環境のインターネットへの接続には家庭用 WiFi ルータを利用した．VPS は US の IP アドレスを持つものを利用した．

4. 観測結果

観測環境を用いて 2018 年 6 月初めから 2018 年 7 月末まで観測を行った結果を報告する．

4.1 観測結果

ボットと C&C サーバとの通信を観測して確認できた内容を示す．

通信データは両方向共に暗号化されている．一部例外もあったがそれについては後段で詳述する．

図 2 は両方向の通信データのデータ構造である．

(a) ボットから C&C サーバへ

(b) C&C サーバからボットへ

図 2: 通信データ構造

両方向ともに，先頭の 4byte はデータの長さを示しており，それに続いて背景を灰色にした部分がデータとして暗号化された状態で保持されている．データ部分の構造は，意味を持つデータが 4byte を基本とした単位で区切られており，リトルエンディアンで保持されている．各通信データのデータ構造の詳細は後述する．

暗号化されたデータを復号するための処理方法は，過去に報告された研究で示されているアルゴリズムから変化していない．今回の解析において復号に成功した鍵は以下のとおりだった．

ボットから C&C への通信を復号する鍵：

`turyqioikleraotsorp␣nehco␣ote`

C&C からボットへの通信を復号する鍵：

`eto␣ochen␣prostoarelkioiqyrut`

続いて，通信シーケンスの各タイミングにおける通信データのデータ構造と，そこに設定される値の変化について説明する．

図 3 がボットと C&C サーバとの通信シーケンスである．

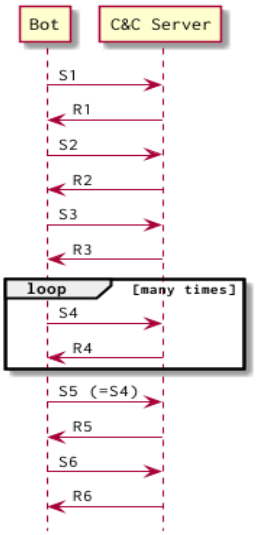

図 3: 通信シーケンス

通信シーケンスの全体を通して通信方向毎のデータ構造に着目すると，それぞれ以下が観測できた．

ボットから C&C サーバへの通信のデータ構造は length に 0 が設定されていても，必ず 32byte の大きさを持っており，length が 0 以外に設定された場合にも，先頭の 32byte のデータ構造は変化せず，32byte の後ろに length で指定された長さのデータが付加されていた．必ず送信されている 32byte はヘッダ情報であると考えることが出来る．

C&C サーバからボットへの通信については，32byte のヘッダ情報は含まれておらず cmdType のみが必須であり，その値に応じてデータ構造が変化していた．

通信シーケンスの S1 において currInstallVer はレジストリの `HKCU\SOFTWARE\Microsoft\OSVersion` というキーに保存された値が設定されていた．localIP はボットとなった PC 自身の持つ IP アドレスが設定されていた．configVer および maillistID は 0 が設定されていた．

R1 において cmdType は 7 であった．このデータ構造を図 4 に示す．この通信データにより C&C サーバからボットに対して設定される情報が送られる事が観測できた．この観測環境で観測できたデータの内容は [2] と異なるが，データ構造は全く同じだった．

length (4byte)	cmdType = 7 (4byte)	configID (4byte)	configVer (4byte)
NULL 文字で区切られた、表示可能なASCII 文字列による設定項目と設定値の羅列 (length-8 byte)			

図 4: 通信データ構造 R1 (cmdType 7)

S2 において configVer を除いた全ての値は S1 と同じであった．configVer は R1 において受信した値が設定されていた．

R2 において cmdType は 5 である．このデータ構造を図 5 に示す．この通信データでは，C&C サーバからボットに対して installVer, externalIP, timestamp が送られている事が確認できた．データ構造は [2] と全く同じであった．

length (4byte)	cmdType = 5 (4byte)	installVer (4byte)	externalIP (4byte)
timestamp (4byte)	0 固定 (4byte)		

図 5: 通信データ構造 R2 (cmdType 5)

S3 において S2 に対して currInstalledVer, settingFlags が変化する．currInstalledVer, settingFlags の変化の仕方は，全て Neo Tan らの報告 [2] と同じであった．

R3 において cmdType は 8 である．このデータ構造を図 6 に示す．この通信データにより，スパムメールの送信元となるメールアドレスの羅列が設定される．データ構造は Neo Tan らの報告 [2] と全く同じであった．

length (4byte)	cmdType = 8 (4byte)	emaillistID (4byte)	(ここから↓のデータ)
NULL 文字で区切られた、スパムメールの送信元として設定されるメールアドレスの羅列 (length-8 byte)			

図 6: 通信データ構造 R3 (cmdType 8)

S4 において emaillistID が変化する．emaillistID の変化の仕方は [2] と同じであった．

R4 においてデータ構造が変化する．このデータ構造を図 7 に示す．これは Neo Tan らの報告 [2] では報告されていないデータ構造である．この R4 のデータは暗号化されておらず，length は 0 に設定されており，その直後に currInstallVer のみが設定されて送られていた．このデータにはこれまでのデータに含まれていたようなボットに指示を与えるような新しい情報が無かった．また，このデータが数回送られた後 TCP のコネクションが切断される事が多かったため，C&C サーバから送る情報がない時に送信されるデータではないかと推測した．

S4 と R4 のやり取りはこの後何度も繰り返される．

S5 のデータ構造と値は S4 と同じである．C&C サーバへ送る通信データは全く同じなのだが，ボットへ送られる通信データが突然 R5 に変化する．これはおそらく，攻撃

length (4byte)	currInstallVer (4byte)

図 7: 通信データ構造 R4

者の指示などにより C&C サーバの条件が整ったタイミングで変化するのだと考えられる．

R5 のデータ構造を図 8 に示す．この通信データによりスパムメールのテンプレートが設定される．このデータ構造は Neo Tan らの報告 [2] と比較して，先頭が vendorID である事は一致しているが，その後スパムメールのテンプレートと思わるデータが始まるまでの 16byte に何かのデータが追加されていた．R5 のデータは観測できた数が少なく，観測できた値を見る限りでは何を示しているか明らかにできなかった．

length (4byte)	cmdType = 6 (4byte)	vendorID (4byte)	data1 = 0 (4byte)
data2 = 0x15 (4byte)	data3 = 1 (4byte)	data4 = 0 (4byte)	(ここから↓のデータ)
スパムメールのテンプレートとなるデータの羅列 (length-20 byte)			

図 8: 通信データ構造 R5 (cmdType 6)

S6 において length に 0x14 が設定されるのを観測した．length に 0 以外が設定されるのはこれが初めてである．またシーケンスのこのタイミングにおいて，ボットから C&C サーバに向けた通信データを復号する鍵が変化する．C&C サーバからボットへの通信データを復号する鍵と同じになり，通信の両方向で同じ鍵を使用することになった．このデータ構造を図 9 に示す．実際のデータも，これまでボットから C&C サーバに向けられていた通信データに加えて 20 (0x14) byte のデータが付加されている事が観測できた．付加されたデータの先頭 4byte は vendorID と同じ値が設定されていた．それ以外のデータは観測できた値を見る限りでは何を示しているか明らかにできなかった．

length = 0x14 (4byte)	currInstalledVer (4byte)	localIP (4byte)	altPort (4byte)	
configVer (4byte)	emaillistID (4byte)	localOSVer (4byte)	settingFlags (2byte)	respSMTPcount (2byte)
vendorID (4byte)	data1 = 1 (4byte)	data2 = 0x10 (4byte)	data3 = 0 (4byte)	
data4 = 0 (4byte)				

図 9: 通信データ構造 S6

R6 のデータ構造を図 10 に示す．この通信データにより，スパムメールの送信先メールアドレスのリストが設定される．データ構造は Neo Tan らの報告 [2] と全く同じであった．

S7 においてボットから C&C サーバに向けて大きなデータが送信された．このデータ構造を図 11 に示す．この通信データは S6 のデータの後に詳細不明なデータが付加さ

length (4byte)	cmdType = 1 (4byte)	vendorID (4byte)	emailEntrySize (4byte)
スパムメールの送信先であるメールアカウントとドメイン名のリスト (emailEntrySize = length-8 byte)			

図 10: 通信データ構造 R6 (cmdType 1)

れた構造となっている．詳細不明なデータの内容はさらなる解析が必要であるが，R6 と S7 の間でボットがスパムメールを送信しているのを観測できたため，スパムメールの送信状況を C&C サーバに伝達するための通信データである可能性が考えられる．

length (4byte)	currInstalledVer (4byte)	localIP (4byte)	altPort (4byte)	
configVer (4byte)	emaillistID (4byte)	localOSVer (4byte)	settingFlags (2byte)	respSMTPcount (2byte)
vendorID (4byte)	data1 = 1 (4byte)	data2 = 0x10 (4byte)	data3 (4byte)	
data4 (4byte)	詳細不明なデータ (length-20 byte)			

図 11: 通信データ構造 S7

4.2 通信プロトコルにおける問題点

通信データの観測を行った際，興味深い現象を観測することが出来た．図 12 は，ある時点の C&C サーバからボットへ向けた通信の暗号化された状態のデータの末尾を示している．

```
0002ab50: bbf1 e4f6 1714 5c0b 1419 001d 0e09 1714  ......\.........
0002ab60: 215d 401b 1b13 70df 919a 9763 6f44 0616  !]@...p....coD..
0002ab70: 4bed cabe eae7 f8ff ffe0 f2f3 e9fb fd8b  K...............
0002ab80: 736f 7270 df91 eef9 fde4 b1e5 e4f9 3419  sorp..........4.
0002ab90: 160b 100e 0808 076c 9a8d 9e90 7473 0a06  .......l....ts..
0002aba0: 1144 4017 0d13 0054 0a11 17f9 e5e9 e7f8  .D@....T........
0002abb0: faf1 e594 6c65 7261 908b e2f1 e2e3 f1e3  ....lera........
0002abc0: ffe7 f3e4 baf5 f9fc 6961 6e73 6538 4066  ........ianse8@f
0002abd0: 7265 6574 6f70 6572 2e72 6576 6965 7700  reetoper.review.
0002abe0: 00ff ffff ff                             .....
```

図 12: 観測できた通信データの問題箇所

これは cmdType 8 の通信データであった．cmdType 8 はスパムメールの送信元として設定されるメールアドレスの羅列が暗号化された状態で保持されているはずだが，図 12 に示したデータの末尾 29 byte を見ると，メールアドレスの一部と解釈できる文字列が並んでいる．

これは暗号化処理の本来の目的であるデータ秘匿の観点からすると，正しく実現できていないように見える．Cutwail ボットネットの暗号処理における問題点であると考えられる．

```
0002ab50: 6f70 6572 2e62 6964 0000 ffff ffff 0061  oper.bid.......a
0002ab60: 7468 3432 4066 7265 6574 6f70 6572 2e61  th42@freetoper.a
0002ab70: 6363 6f75 6e74 616e 7400 00ff ffff ff00  ccountant.......
0002ab80: 6d65 6469 6174 696f 6e69 6c33 4066 7265  mediationil3@fre
0002ab90: 6574 6f70 6572 2e64 6174 6500 00ff ffff  etoper.date.....
0002aba0: ff00 6c61 6767 6172 646c 4066 7265 6574  ..laggardl@freet
0002abb0: 6f70 6572 2e6c 6f61 6e00 00ff ffff ff00  oper.loan.......
0002abc0: 7361 6c76 6164 6f72 969e 918c 9ac7 bf99  salvador........
0002abd0: 8d9a 9a8b 908f 9a8d d18d 9a89 969a 88ff  ................
0002abe0: ff00 0000 00                             .....
```

図 13: 問題箇所を復号した状態

今回の解析では暗号アルゴリズムとして，Neo Tan らによって報告 [2] された疑似コードを実装したものと，Genki Saito の報告 [3] に添付された python のコードを利用したものを実装し処理を行った．しかし，どちらも問題箇所の復号処理を行った際に図 13 の状態となってしまった．図 12 で示したメールアドレスの一部と解釈できる箇所が，人間に読み取れない状態に変化した事が確認できる．

通信データが図 12 の状態となった時にも，ボットが受け取った全てのデータを正しく解釈できる状態に戻すことが出来るよう，今回の解析では暗号アルゴリズムの修正を行った．

修正するための原因の調査に当たって，データの長さに着目した．今回の解析で復号処理に使用した暗号アルゴリズムは，過去の研究で報告されているとおり，29byte の鍵を用いて，29byte を一つのブロックとして，29byte 毎に繰り返し処理をするものである．図 12 のメールアドレスの一部と解釈できる文字列の先頭から通信データの末尾までの長さは 29byte である事が確認できる．また，この通信データに含まれる暗号対象データの長さは 175073byte であり，175073 は 29 の整数倍であった．

これらの事から，暗号対象データの長さが 29 の整数倍の時に暗号対象データを最後まで暗号化出来ないアルゴリズムになっている可能性があると推測し，暗号アルゴリズムの一部を修正した．

Neo Tan らの報告 [2] にあった疑似コードを元に，必要最小限のみ変更した形でこの問題に対処した暗号アルゴリズムの疑似コードを図 14 に示す．

修正した暗号アルゴリズムを実装し，問題の通信データを復号したところ，図 15 に示すとおり，問題の箇所についてもメールアドレスとして解釈できる状態に復号できていることが確認できた．

4.3 過去の報告との比較

Cutwail についてこれまでに報告されている研究の結果と，今回観測できた結果を比較する．

通信データのデータ構造，暗号鍵および暗号アルゴリズムに着目して分類を行った．以下の表 2 で示す．

表 2: 過去の研究との比較

報告者	データ構造	暗号鍵	暗号アルゴリズム
Alice Decker ら [1]	同一	同一	同一
Genki Saito [3]			
Neo Tan ら [2]	同一	同一	
(本稿)		(注)	(一部修正)

Alice Decker らの報告と Genki Saito の報告は，データ構造，暗号鍵，暗号アルゴリズムの全てが同一であった．

```
total = 0;
keyLength = 0x1D;
//when receiving it uses 'turyqioikleraotsorp nehco ote'
sKey[keyLength] = 'eto ochen prostoarelkioiqyrut';
j = 0;
while (dataSize != 0) {
  if (dataSize == keyLength) {
    break;
  } else if (dataSize <= keyLength) {
    data[total] =~ data[total]; //not operation
    total++;
    dataSize--;
  } else {
    j = 0;
    k = total;
    //xor key string backwards
    for (j = 0; j < keyLength; j++) {
      temp[j] = sKey[keyLength-1-j] ^ data[total];
      total++;
    }
    total = k;
    //reverse result
    for (j = 1; j <= keyLength; j++) {
      data[total] = temp[keyLength - j];
      total++;
    }
    if ((k&1) != 0) { //total is odd
      total = k;
      for (j = 0; j <keyLength; j++) {
        data[total] =~ data[total]; //not operation
        total++;
      }
    }
    dataSize -= keyLength;
  }
}
```

図 14: 復号処理の疑似コード

```
0002ab50: 6f70 6572 2e62 6964 0000 ffff ffff 0061  oper.bid.......a
0002ab60: 7468 3432 4066 7265 6574 6f70 6572 2e61  th42@freetoper.a
0002ab70: 6363 6f75 6e74 616e 7400 00ff ffff ff00  ccountant.......
0002ab80: 6d65 6469 6174 696f 6e69 6c33 4066 7265  mediationil3@fre
0002ab90: 6574 6f70 6572 2e64 6174 6500 00ff ffff  etoper.date.....
0002aba0: ff00 6c61 6767 6172 646c 4066 7265 6574  ..laggardl@freet
0002abb0: 6f70 6572 2e6c 6f61 6e00 00ff ffff ff00  oper.loan.......
0002abc0: 7361 6c76 6164 6f72 6961 6e73 6538 4066  salvadorianse8@f
0002abd0: 7265 6574 6f70 6572 2e72 6576 6965 7700  reetoper.review.
0002abe0: 00ff ffff ff                             .....
```

図 15: 修正した処理により問題箇所を復号した状態

Neo Tan らの報告は，上記二つの報告とはデータ構造と暗号鍵が異なっていた．今回我々が観測したものは Neo Tan らによる報告と，データ構造と暗号鍵が同一だった．

暗号鍵については，Neo Tan らによる報告を表記通りに読み取ると復号することが出来なかったが，二箇所に空白文字を入れることで意味のあるデータとして解釈できる形に復号することが出来たため，これは表記の問題であり，観測した Cutwail が使用していた暗号鍵と Neo Tan らにより報告された暗号鍵は同一であると結論づけた．また，通信の方向に応じて暗号鍵のデータの並びを逆順にするところも過去に報告されているとおりだった．

暗号アルゴリズムについては，今回の解析において先に記述したとおり修正したが，基本的なアルゴリズムは同一であるため，全て同一であると結論づけた．おそらく，過去の報告では 29 の整数倍となる通信データが観測できなかったため記述されていなかったのだと考えられる．

観測できた通信データのデータ構造の中で，R4 だけは Neo Tan らの報告には記載が無かったが，その他のデータ構造，暗号鍵，暗号アルゴリズムが全て同じであったため，今回観測した個体は Neo Tan らに報告されたものとごく近い亜種であると考えられる．今回の観測により，2013 年に報告されていた Cutwail の通信プロトコルが，2018 年現在でも引き続き使用されている事が確認できた．

5. 今後の課題

Cutwail のボットと C&C サーバの間の通信シーケンスのうち，主要な通信データについては設定値の意味を把握することが出来たが，観測できた情報だけでは推測が難しい通信データもあった．これらについても詳細な解析を進めるため，引き続き Cutwail の通信データの観測と解析が必要である．

今回 Cutwail のボットと C&C サーバの間の通信データを観測し詳細に解析したことにより，29 の整数倍のデータの時に最後の 29byte が暗号化されない状態になっているという暗号アルゴリズムの不具合を発見できた．ボットや C&C サーバの処理の中にはこの他にも不具合が残っている可能性がある．不具合の内容次第で，不具合を利用したボットネットの停止や縮小などが出来る可能性もあるため，さらなる観測と詳細な解析を続けたい．

参考文献

[1] Alice Decker, David Sancho, Loucif Kharouni, Max Goncharov, Robert McArdle: A study of the Pushdo / Cutwail Botnet, 入手先 ⟨https://www.trendmicro.de/cloud-content/us/pdfs/business/white-papers/wp_study-of-pushdo-cutwail-botnet.pdf⟩ Trend Micro (2009).

[2] Neo Tan, He Xu, Kyle Yang: Pushdo's new second generation, 入手先 ⟨https://www.virusbulletin.com/virusbulletin/2013/04/pushdos-new-second-generation⟩ Virus Bulletin (2013).

[3] Genki Saito: Analysis of the Cutwail Botnet Command and Control Software, University College London Department of Computer Science (2015).

[4] 岡本勝之: 国内ネットバンキングを狙うマルウェアスパムボットネットに「潜入調査」, 入手先 ⟨https://blog.trendmicro.co.jp/archives/14538⟩ Trend Micro (2017).

[5] 岡本勝之: 国内ネットバンキングを狙う新たな脅威「DreamBot」を解析, 入手先 ⟨https://blog.trendmicro.co.jp/archives/14588⟩ Trend Micro (2017).