Малварь со странностями. Разбираем ключевые фрагменты кода малвари на скриптах и средствах автоматизации

Евгений Дроботун, 01.11.2014 · Комментарии · 706 · Добавить в закладки

Содержание статьи

Мы уже давно не заикаемся об ассемблере, поскольку привыкли к тому, что большая часть современной малвари пишется на C++, C# или даже VB. Однако из антивирусных компаний передают, что вирмейкеры XXI века пользуются не только классическими языками программирования. Оказывается, вполне злая и функциональная малварь теперь пишется на батниках, AutoIt, Lua, Python, 1C... Сегодня мы попробуем заглянуть под капот этому «программному обеспечению» и рассмотреть ключевые, ответственные за главный функционал участки кода.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

BAT-СКРИПТЫ

Если ты думаешь, что единственный возможный вариант такого рода малвари — это классический школьный батник со строкой format c: внутри, то ты ошибаешься.

Возможность автоматизировать всякие рутинные операции в системе с помощью bat-скриптов уже давно переросла в целое направление создания малвари, для которой почти все антивирусные компании отвели целый раздел в своих классификациях вредоносного программного обеспечения.

К примеру, с помощью команды ftp можно загрузить из сети нужный файл, сохранить его в нужном месте и запустить на выполнение, а также добавить этот файл в автозагрузку, написав внутри что-нибудь вроде этого:

```
@reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run"
        /v Trojan /t REG_SZ /d C:\Trojan.bat /f
```

В итоге получается простейший Bat.Downloader, который исправно будет выполнять свои функции.

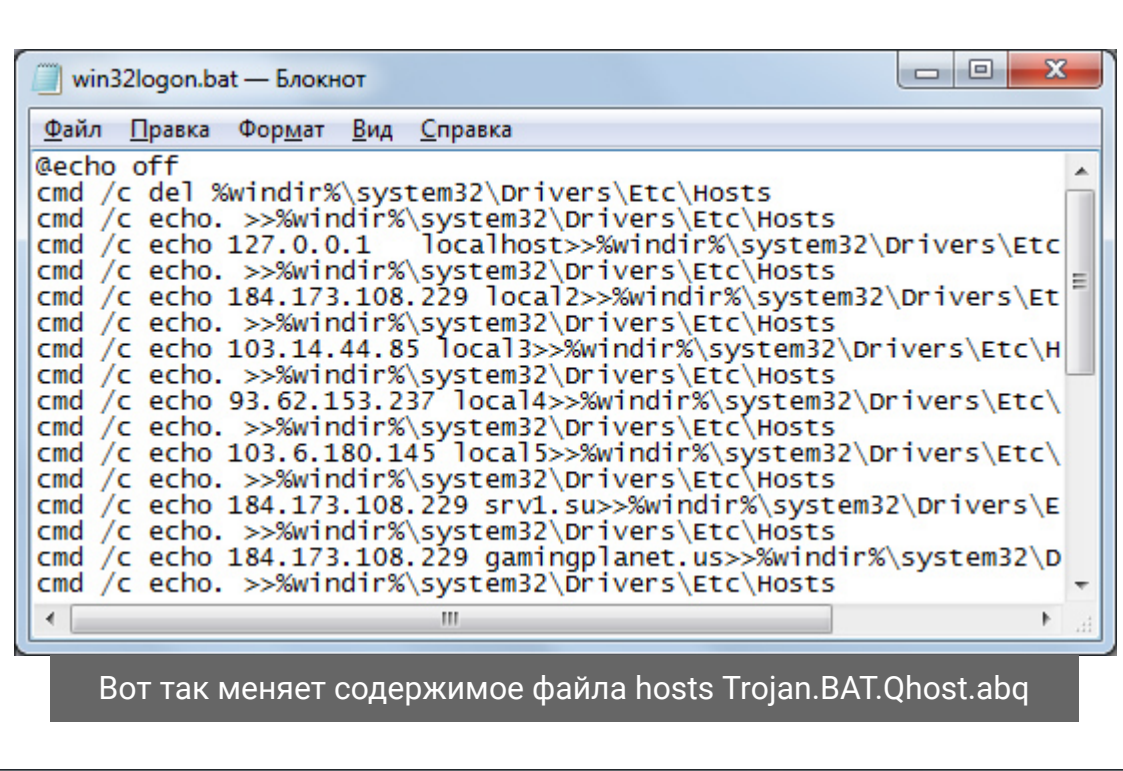
Помимо этого, с помощью команды taskkill можно попытаться остановить работающие процессы мешающих вредоносному функционалу программ:

```
// Убиваем explorer.exe
@taskkill /im explorer.exe /f > nul
// Убиваем некий антивирус aver.exe. С большой вероятностью не получится ;)
@taskkill /im aver.exe /f > nul
```

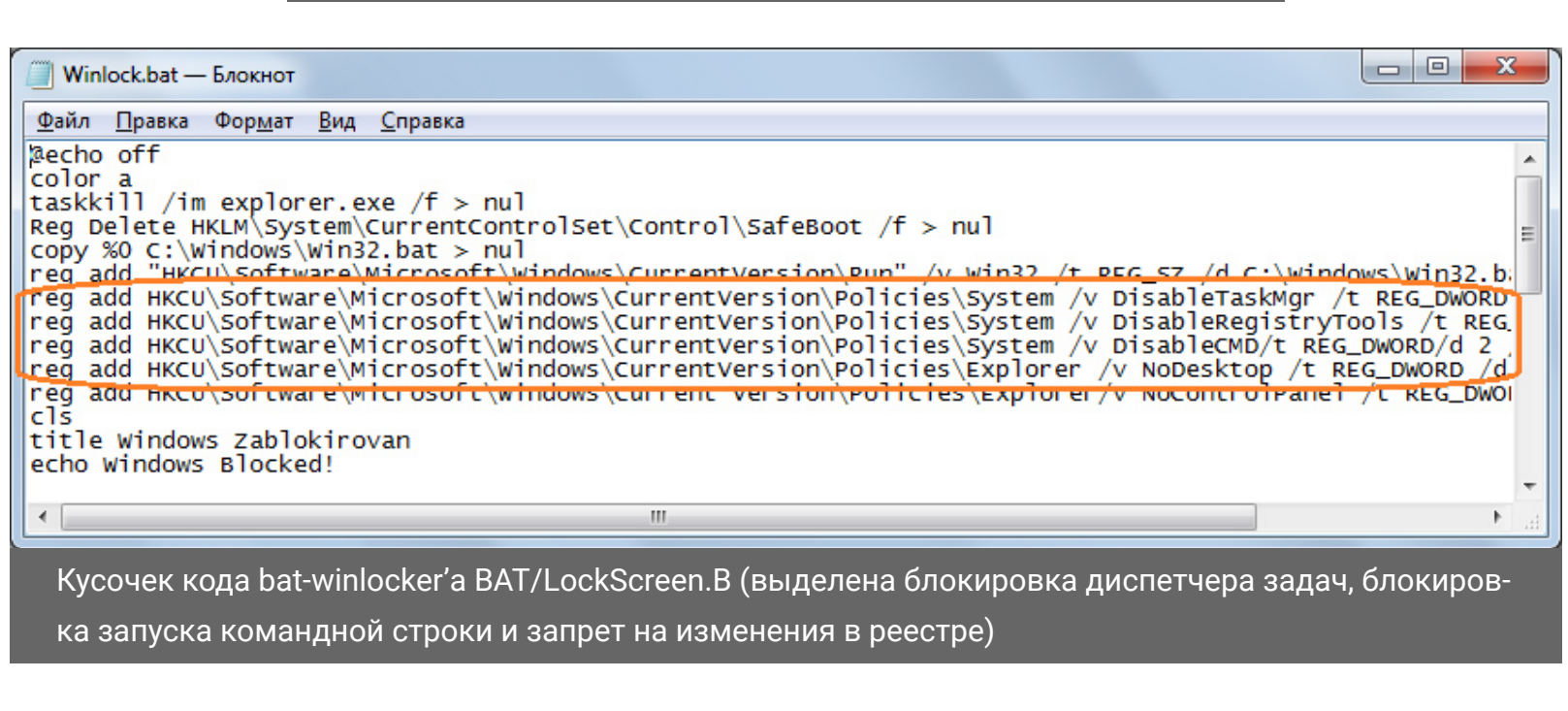
Обычно малварь перед началом своей деятельности проверяет наличие в системе антивирусов и далее действует по результатам. В bat-скрипте это можно сделать с помощью команды tasklist:

```
// Проверяем наличие процесса aver.exe
@for /F "delims=" %%A in ('tasklist /FI "imagename eq aver.exe"')
    do @set sr=%%A
@if "%sr:~,11%"=="aver.exe" goto ff
...
    // Процесс обнаружен
    // Выполняем соответствующие действия
...
@goto bb
:ff
...
    // Процесс не обнаружен
    // Выполняем соответствующие действия
...
:bb
...
    // Работаем дальше
```

Что касается непосредственно вредоносных действий, то здесь имеется достаточно широкое поле для деятельности: можно удалять или перемещать различные системные файлы, изменять содержимое конфигурационных файлов (в том числе и файла hosts), изменять значения в реестре, блокируя тем самым, например, возможность вызова taskmgr.exe или запрещая вносить изменения в реестр.

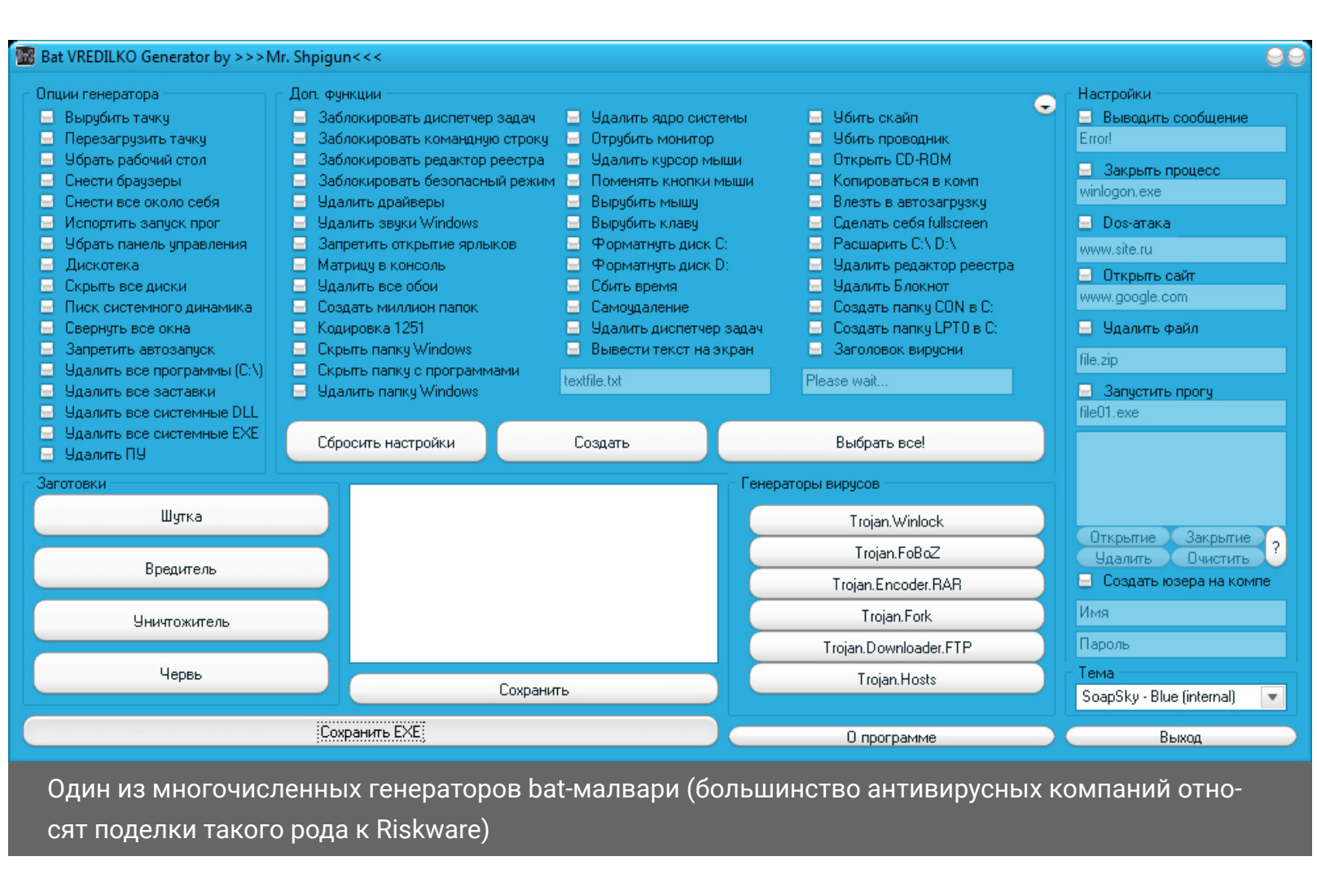


Вот так меняет содержимое файла hosts Trojan.BAT.Qhost.abq



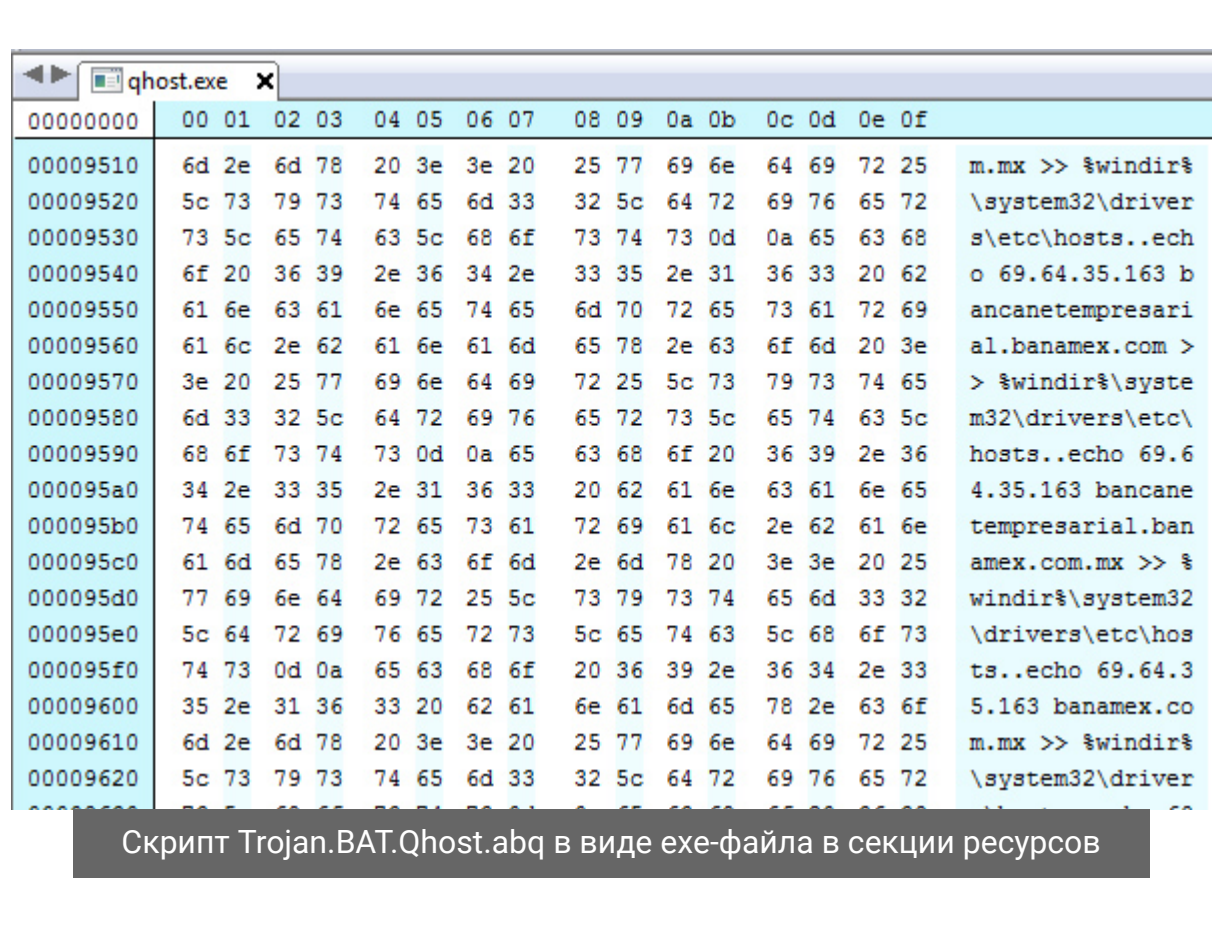
Кусочек кода bat-winlocker'а BAT/LockScreen.B (выделена блокировка диспетчера задач, блокировка запуска командной строки и запрет на изменения в реестре)

В сети можно найти большое количество генераторов bat-малвари. Конечно, всерьез воспринимать все то, что выходит из-под пера этих генераторов, вряд ли стоит, но для не слишком искушенных пользователей даже остановка explorer.exe может стать неразрешимой проблемой.



Один из многочисленных генераторов bat-малвари (большинство антивирусных компаний относят поделки такого рода к Riskware)

Для приведения вредоносных скриптов в более привычный вид исполняемого файла создатели этих скриптов нередко пользуются утилитами конвертации типа bat2exe. Это работает, хотя ни о какой полноценной компиляции здесь говорить не приходится: сам скрипт пишется в exe-файл в виде ресурсов, а затем вызывается cmd.exe, где скрипт и выполняется.

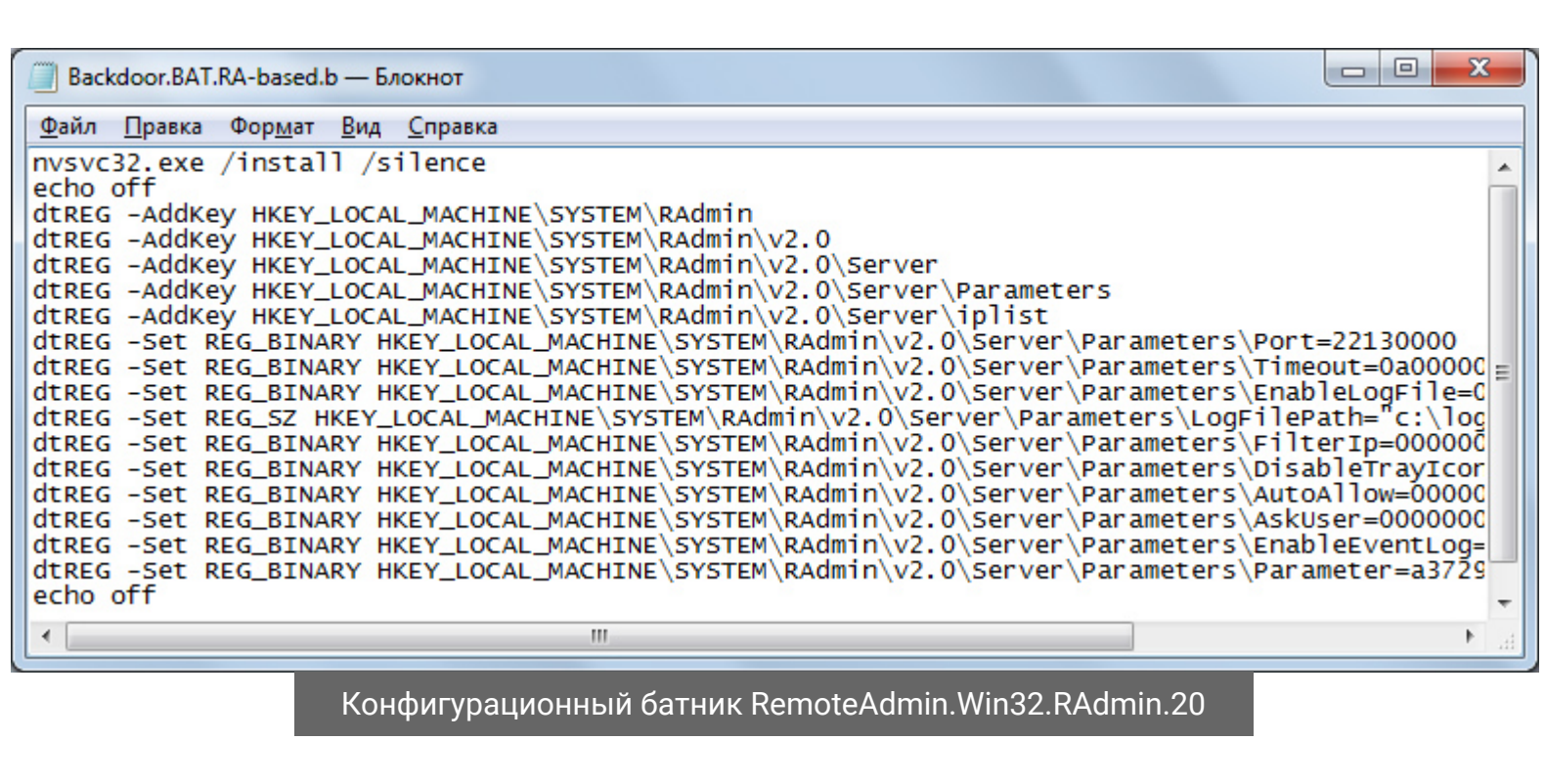


Скрипт Trojan.BAT.Qhost.abq в виде exe-файла в секции ресурсов

САМОРАСПАКОВЫВАЮЩИЕСЯ АРХИВЫ

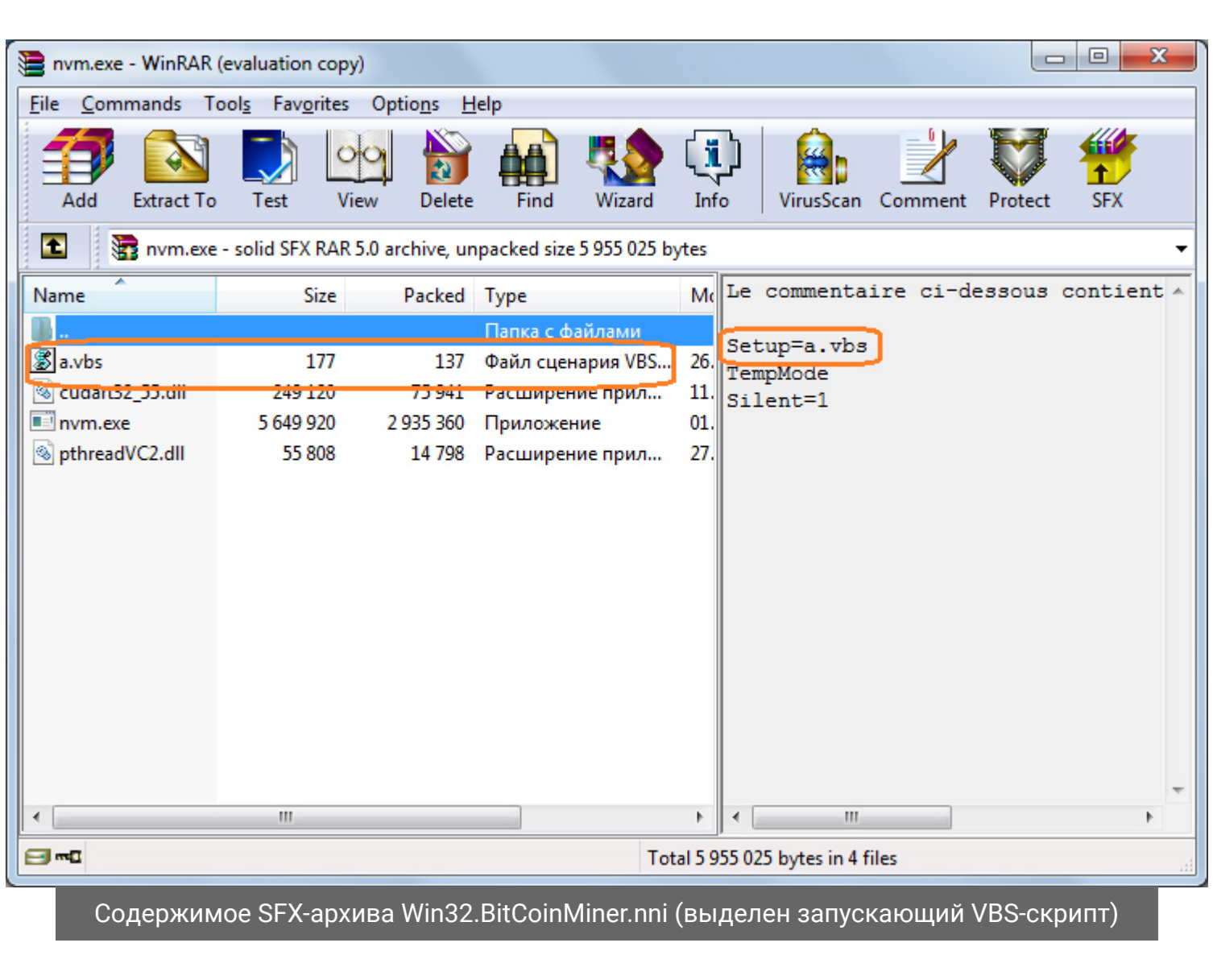
Self-extracting archive (SFX-архивы), а точнее возможность сложить несколько файлов в один архив с автоматическим запуском одного из них после распаковки уже давно приглянулись распространителям сомнительного софта и контента. Зачастую в эти архивы кладут вполне легальные программы вместе с конфигурационным или запускающим батником (или скриптом на VBS), который позволяет использовать такие архивы с не совсем благими намерениями.

К примеру, SFX-архив RemoteAdmin.Win32.RAdmin.20 содержит в себе серверный модуль широко известной утилиты удаленного администрирования Radmin и bat-скрипт, с помощью которого производится конфигурирование и скрытый запуск серверного модуля Radmin. Свою конфигурацию Radmin хранит в реестре, и скрипт, находящийся в архиве, перед запуском Radmin'а прописывает необходимые параметры в нужных ветках реестра.



Конфигурационный батник RemoteAdmin.Win32.RAdmin.20

С появлением в широком обиходе различных криптовалют SFX-архивы полюбились и многим желающим помайнить цифровую наличность на чужих вычислительных мощностях. В большинстве случаев утилиты для майнинга состоят из нескольких файлов, и самораспаковывающийся архив как нельзя лучше подходит для того, чтобы все эти файлы туда спрятать. Разумеется, в архив помещается bat- или VBS-скрипт, запускающий все это дело с нужными параметрами.



Содержимое SFX-архива Win32.BitCoinMiner.nni (выделен запускающий VBS-скрипт)

В некоторых SFX-архивах такого рода также содержится утилита Hidden Start, которая обеспечивает скрытый запуск основной программы вредоносного архива.

AUTOIT

Изначально AutoIt предназначался для автоматизации и выполнения часто повторяющихся задач (например, инсталляции софта на большое количество компьютеров). В более поздних версиях язык обрел черты большинства распространенных языков общего назначения.

INFO

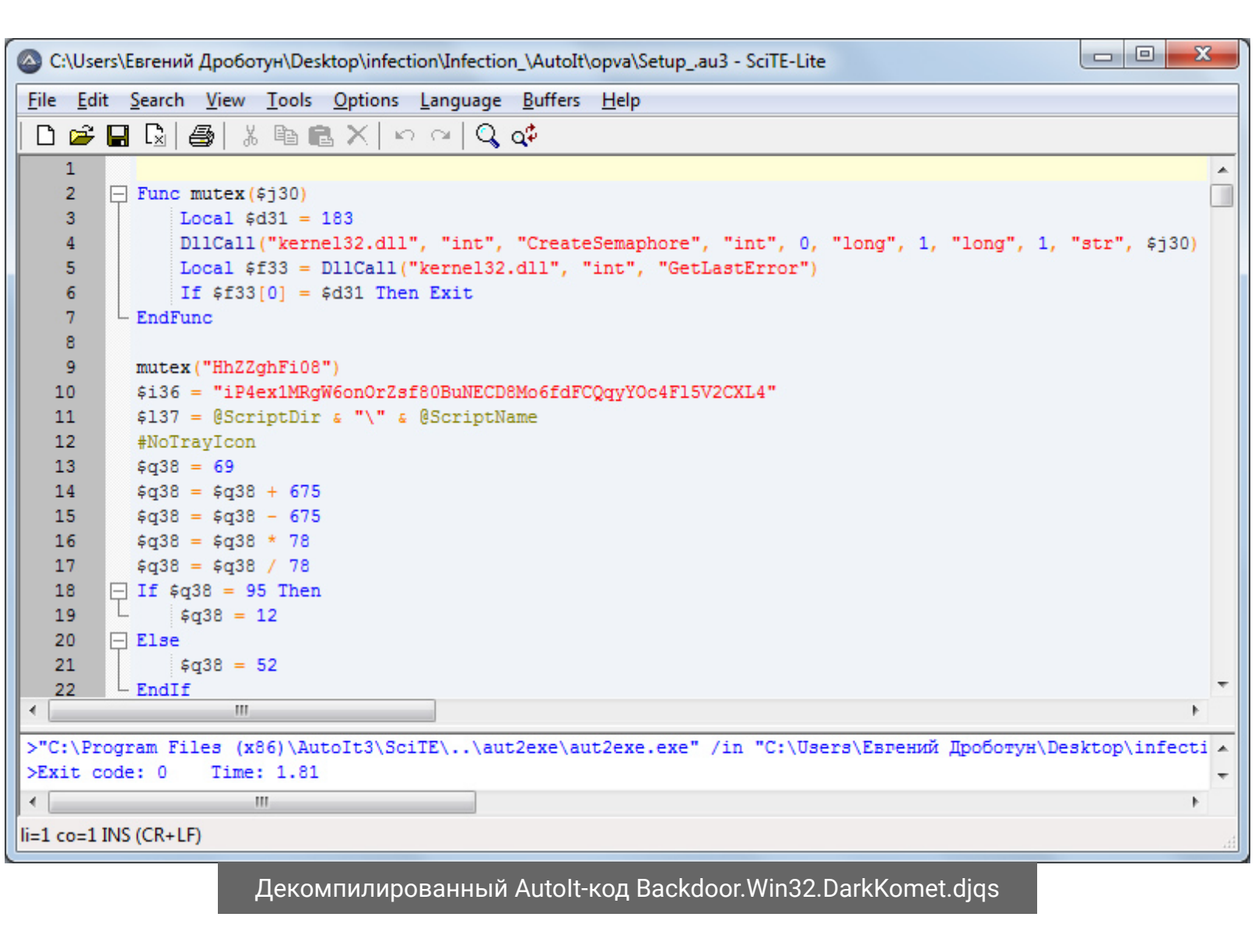
В составе Windows (начиная с Windows XP) имеется утилита IExpress, позволяющая создавать CAB-архивы, в том числе и самораспаковывающиеся.

В настоящее время на AutoIt пишется довольно большое количество малвари. Язык позволяет взаимодействовать с системой на достаточно низком уровне и поддерживает вызов API-функций. Например, чтобы внедрить свой код в какой-нибудь процесс, нужно написать нечто вроде:

```
// Разрешения для OpenProcess
Local $PERMISSION =
    BitOR(0x0002, 0x0400, 0x0008, 0x0010, 0x0020)
...
...
$hProcess = DllCall("kernel32.dll", "ptr", "OpenProcess",
    "dword", $PERMISSION, "int", 0,
    "dword", $Process)
...
...
DllCall("kernel32.dll", "int", "WriteProcessMemory",
    "ptr", $hProcess, "ptr", $pMem, "ptr", $buffer,
    "uint", 260, "uint*", 0)
...
...
DllCall("kernel32.dll", "ptr", "CreateRemoteThread",
    "ptr", $hProcess, "ptr", 0, "uint", 0, "ptr", $pMem,
    "ptr", $pMem, "dword", 0, "ptr", 0)
```

Помимо всех возможностей этого языка, большим плюсом для создателей малвари выступает возможность обфускации кода при его компиляции. Для этого достаточно в начале программы написать две строки вида:

```
// Запустить обфускатор перед компиляцией
#Autoit3Wrapperr_Run_Obfuscator=y
// Установить параметры обфускации
#Obfuscator_Parameters=/StripOnly /OM
```



Декомпилированный AutoIt-код Backdoor.Win32.DarkKomet.djqs

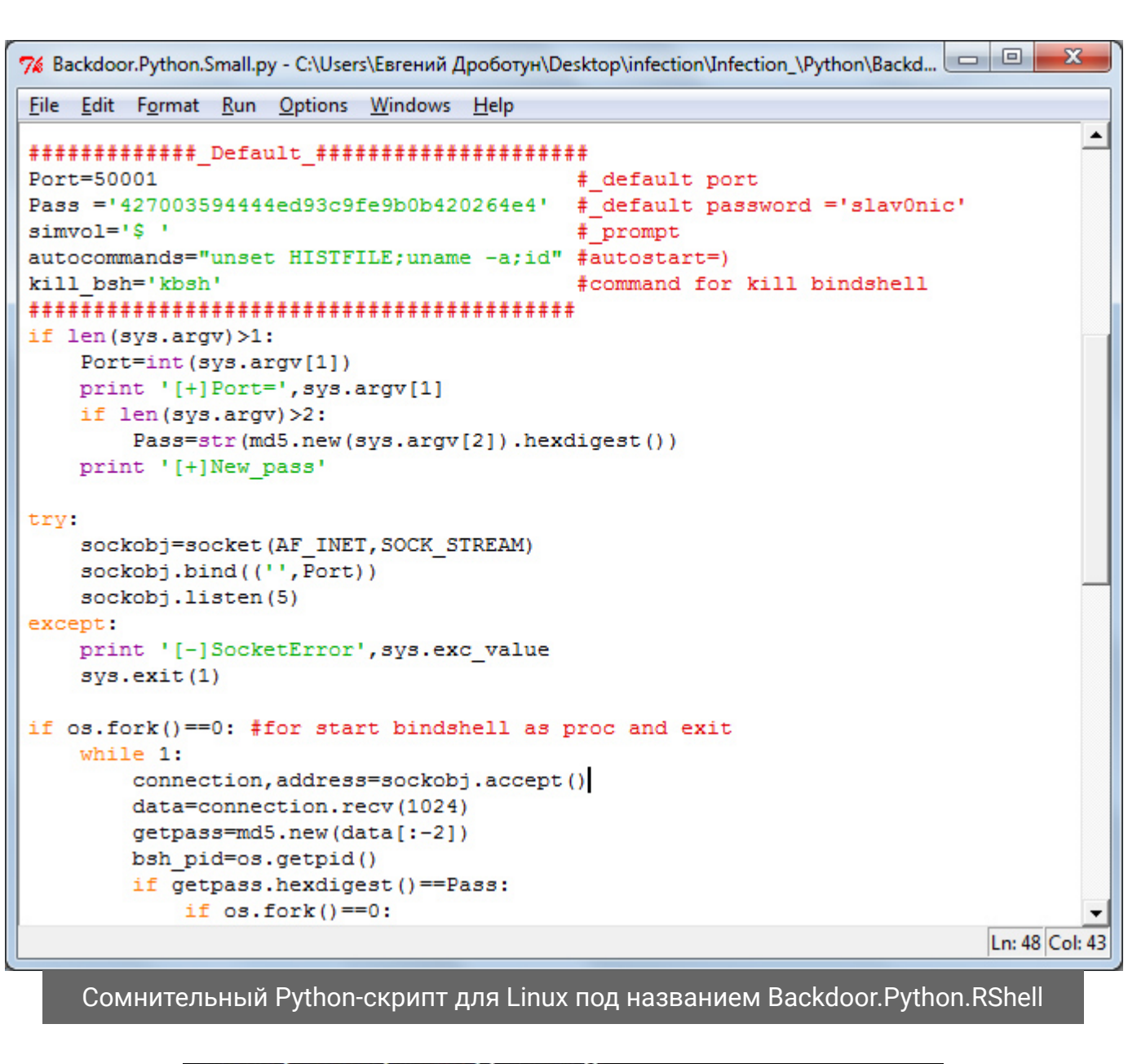
В целом, если не принимать во внимание объем создаваемого компилятором AutoIt исполняемого кода, язык неплохо справляется с большинством задач, которые ставят перед собой вирусописатели. Нередко попадаются весьма продвинутые экземпляры, использующие разные техники внедрения и сокрытия кода, шифрование тела малвари и прочие хитрости.

WWW

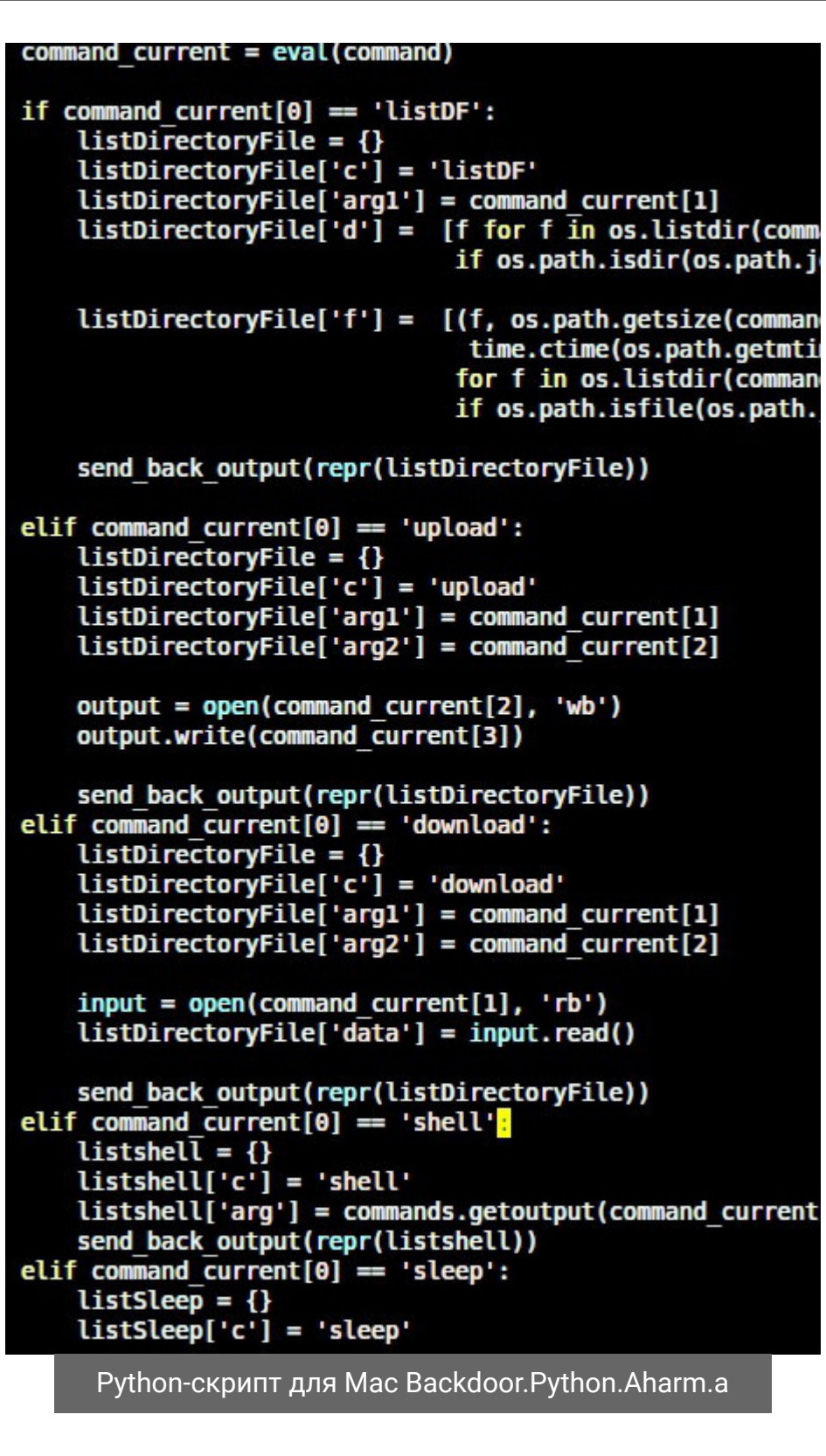
Для AutoIt существует визуальный редактор графического интерфейса, похожий на Delphi, — Koda FormDesigner. Познакомиться с ним можно здесь.

PYTHON, LUA И ПРОЧАЯ ЭКЗОТИКА

Несмотря на то что Python — настоящий хакерский язык программирования, малварь на нем под винду встречается не очень часто. В большей степени написание ее оправданно для OS X или Linux, в которых Python установлен вместе с системой.



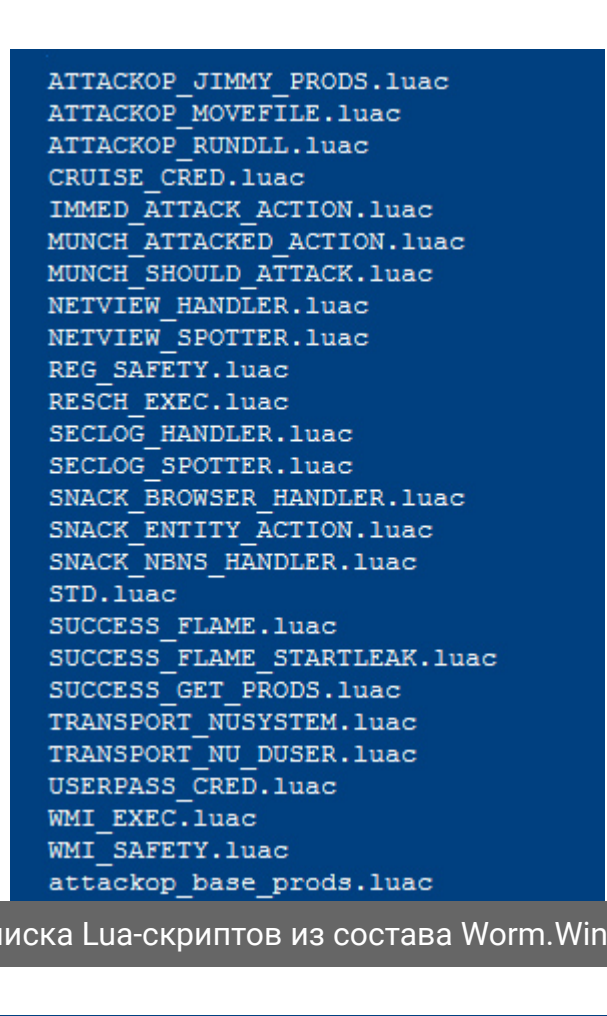
Сомнительный Python-скрипт для Linux под названием Backdoor.Python.RShell



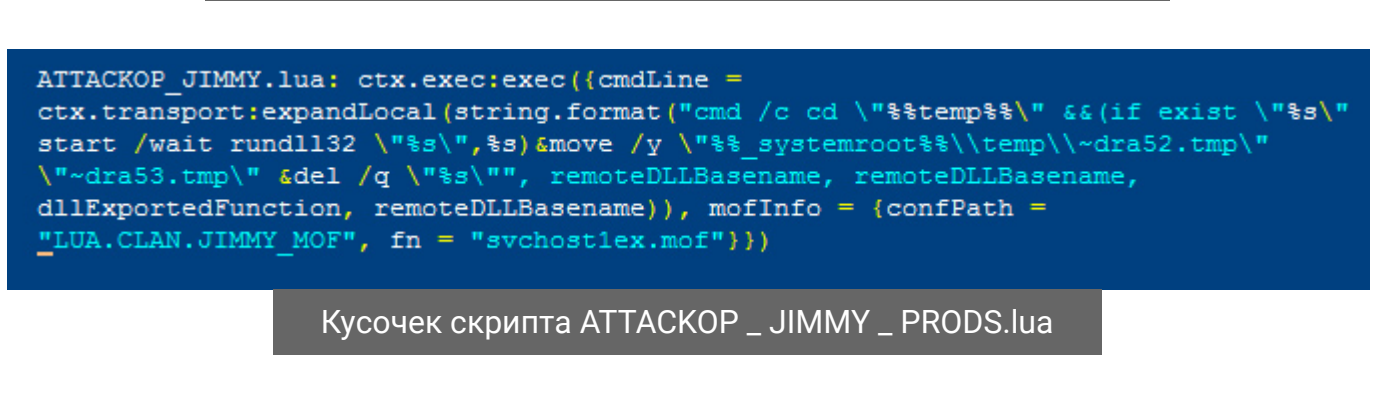
Python-скрипт для Mac Backdoor.Python.Aharm.a

Для винды вредоносные Python-скрипты, как правило, компилят в исполняемый файл (на самом деле это тоже не полноценная компиляция — в exe-файл кладется сам скрипт и интерпретатор Python'а).

Что касается языка программирования Lua, то самым известным вредоносом, который был написан с его использованием, был Worm.Win32.Flame. Для большей части компонентов этого червя логика верхнего уровня реализована именно на Lua. Всего в Worm.Win32.Flame можно насчитать 57 Lua-компонентов, каждый из которых выполняет какую-либо вредоносную функцию. К примеру, скрипт ATTACKOP _ JIMMY _ PRODS.lua производит атаку на другой ПК, скрипт с названием casafety.lua служит для обнаружения антивирусного ПО, CRUISE _ CRED.lua — для кражи учетных данных, а euphoria.lua эксплуатирует уязвимость в LNK-файлах.



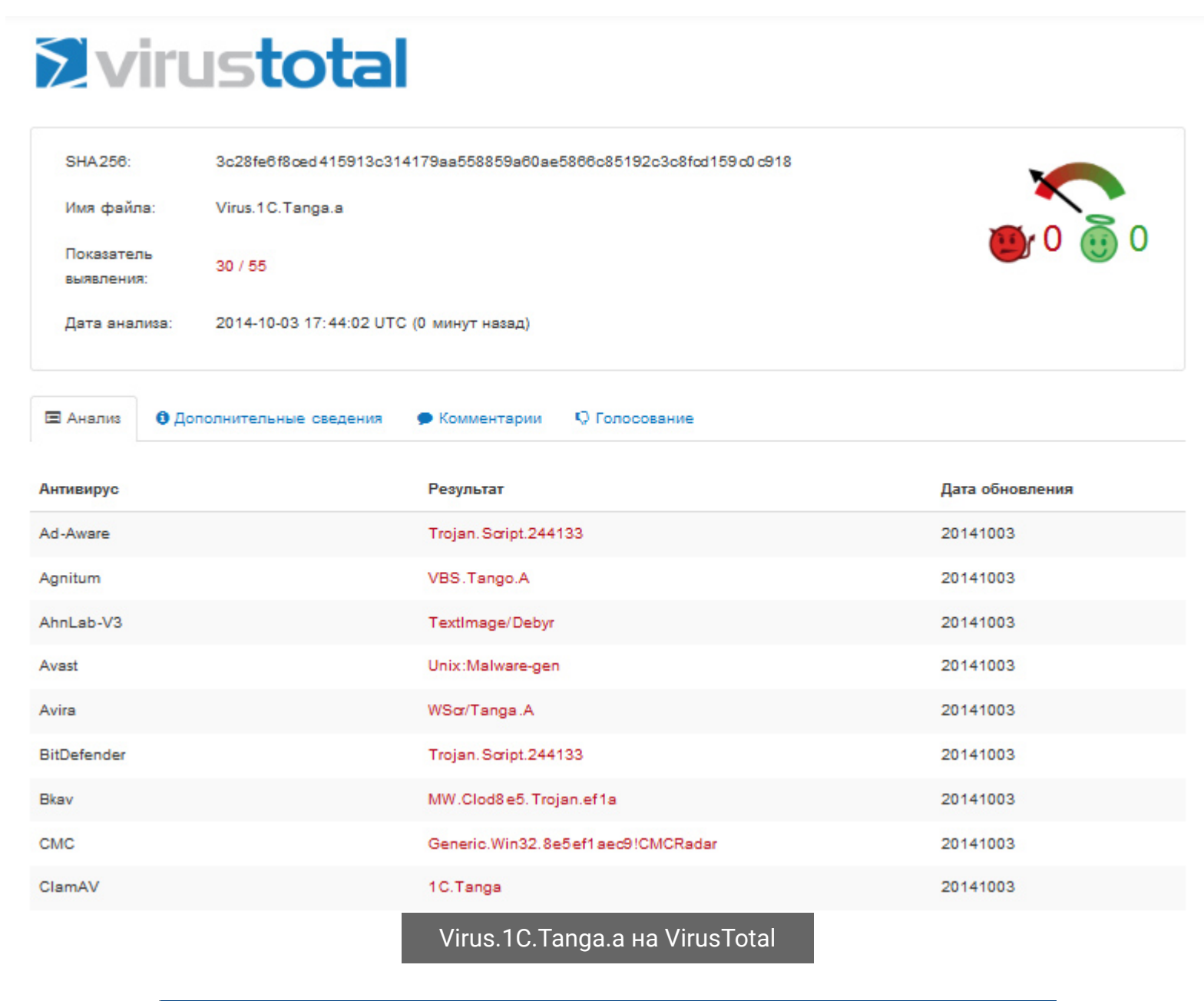
Часть списка Lua-скриптов из состава Worm.Win32.Flame



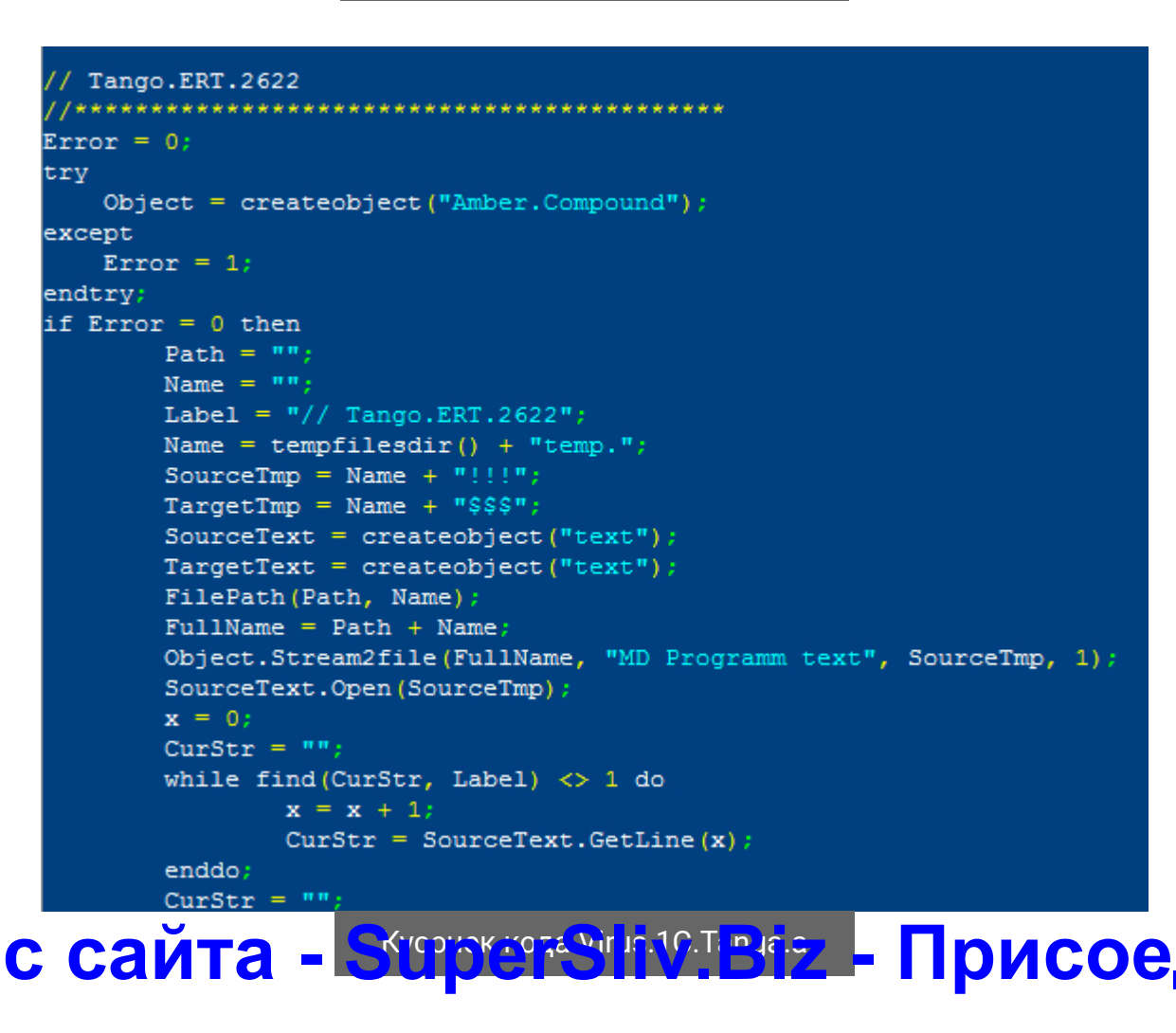
Кусочек скрипта ATTACKOP _ JIMMY _ PRODS.lua

ЗАКЛЮЧЕНИЕ

Как показывает повседневная практика вирусных аналитиков, вредоносный код можно написать на чем угодно, и в коллекциях сэмплов вредоносного кода многих антивирусных компаний встречаются весьма экзотические образцы, написанные, к примеру, на встроенном языке программирования системы «1С:Предприятие» (Virus.1C.Bonny.a, Virus.1C.Bonny.b или Virus.1C.Tanga.a). Мы можем называть перечисленные в статье поделки странными и нетипичными, но они есть, они работают, а «1С:Предприятие», как известно, установлено на очень большом количестве компьютеров нашей родины...



Virus.1C.Tanga.a на VirusTotal



Кусочек кода Virus.1C.Tanga