

CTI elemzés

Hozzáférési malware-ek



NEMZETI
KIBERBIZTONSÁGI
INTÉZET

Tartalomjegyzék

Bevezetés	4
A trójai működés technikai logikája	5
A trójai mint működési modell	5
Látható működés és háttérben futó technikai folyamatok	5
Inicializálás és környezetfelmérés	7
Inicializálás és környezetfüggő viselkedés	7
Környezetellenőrzés és elemzői környezet felismerése	8
Dropperek és loaderek szerepe	10
A dropper helye a hozzáférési láncban	10
Többlépcsős payload-kibontás	11
Memóriában történő betöltés	12
Dropper és loader kapcsolata	12
Backdoorok: rejtett hozzáférési pontok	14
A backdoor szerepe a hozzáférés megtartásában	14
A visszatérés technikai feltételei	15
Backdoorra utaló technikai nyomok	16

RAT-ok: távoli vezérlés technikai modellje	18
A RAT szerepe a hozzáférési láncban	18
A fertőzött oldali komponens működése	18
Távoli vezérlési képességek	19
Komponensek kezelése és további műveletek előkészítése	20
Botnet kliensek és vezérlési logika	22
A botnet kliens szerepe a fertőzött rendszerben	22
Parancsok kiosztása és végrehajtása	22
Mire használható egy botnet kliens?	23
C2 kommunikáció és hozzáférésfenntartás	24
A C2 kapcsolat szerepe	24
Kapcsolatfelvétel és bejelentkezés	24
Parancslekérés és válaszlogika	25
Titkosítás és a forgalom elrejtése	26
Tartós jelenlét és jogosultsági szint	27
Automatikus újraindítási pontok	27
Jogosultsági környezet és futási szint	27
Összegzés	30
Források	32

Bevezetés

Jelen dokumentum célja a **trójai és hozzáférési malware-ek technikai működésének bemutatása**. Az elemzés azt vizsgálja, hogyan alakulhat ki egy kezdeti kódfuttatásból rendszerszintű jelenlét, távoli hozzáférés vagy vezérlési lehetőség. Ennek részeként bemutatásra kerülnek a trójaiak, RAT-ok, backdoorok, loaderok, dropperek és botnet kliensek főbb technikai jellemzői, valamint az a szerep, amelyet a hozzáférés kiépítésében és fenntartásában betölthetnek.

A trójai és hozzáférési malware-ek sajátossága, hogy működésük középpontjában nem feltétlenül az önálló terjedés vagy a közvetlen károkozás áll. Technikai szempontból sokkal inkább olyan **előkészítő vagy hozzáférést biztosító komponensek** értelmezhetők, amelyek a **támadási lánc korai szakaszában** jelennek meg.

A kezdeti fertőzés nem csak abból áll, hogy a felhasználó vagy a rendszer kapcsolatba kerül egy kártékony kóddal. A folyamat része az is, hogy a kód milyen módon kerül be a rendszerbe, és milyen lépések történnek a futtatása során. Ide tartozhat a futási környezet vizsgálata, a jogosultsági szint felmérése, a konfiguráció betöltése, további komponensek előkészítése, valamint a támadói infrastruktúrával történő kommunikáció kialakítása. Ezek a műveletek határozzák meg, hogy a fertőzés egyszeri esemény marad, vagy **hozzáférési ponttá válik** későbbi utasításokra.

Az elemzés ezeknek megfelelően a hozzáférés kialakulásának technikai logikájára helyezi a hangsúlyt. A cél annak bemutatása, hogy a kezdeti kódfuttatás miként válhat **tartósabb jelenlétté**, hogyan jöhet létre távoli vezérlési lehetőség, és milyen komponensek játszhatnak szerepet abban, hogy a támadó a kompromittált rendszert későbbi műveletek kiindulópontjaként használja.

A trójai működés technikai logikája

A trójai mint működési modell

A trójai malware-ek technikai értelmezésekor fontos kiindulópont, hogy a „trójai” megnevezés nem egyetlen konkrét megvalósítási formát jelöl. Sokkal inkább egy működési modellt ír le, amelynek központi eleme a látszólagos és a valós funkcionalitás közötti eltérés. Egy trójai jellegű fájl vagy egy futtatható komponens **olyan benyomást kelthet, mintha legitim, hasznos vagy ártalmatlan célt szolgálna, miközben a háttérben olyan műveleteket hajt végre**, amelyek a rendszer kompromittálását vagy a támadói hozzáférést segítik elő.

Technikai szempontból a trójaiaknál nem az önálló terjedés a meghatározó elem. A hangsúly arra helyeződik, hogy a célrendszeren létrejön egy kezdeti futtatási lehetőség, amelyből majd további műveletek hajthatók végre. Ez lehet környezetfelmérés, konfiguráció betöltése, új komponens előkészítése, hálózati kommunikáció indítása vagy egy későbbi hozzáférési funkció megalapozása. A trójai tehát gyakran nem a támadás végső célját valósítja meg, hanem azt a technikai állapotot hozza létre, amelyből a támadó később továbbléphet.

Látható működés és háttérben futó technikai folyamatok

A trójai működés egyik alapvető sajátossága, hogy a látható működés és a háttérben zajló technikai folyamatok nem feltétlenül esnek egybe. A felhasználó oldalán megjelenhet egy telepítő, egy dokumentum, egy hibaüzenet vagy akár egy valóban működő alkalmazásrész is. Ettől függetlenül a háttérben elindulhatnak olyan rendszerinterakciók, amelyek nem indokolhatók a látható funkcióval.

Ilyen **háttérművelet** lehet például a rendszerinformációk lekérdezése, a futtatási környezet ellenőrzése, fájlok létrehozása, beágyazott adatok kibontása, konfigurációs értékek beolvasása, folyamatok indítása vagy hálózati kapcsolat előkészítése. Ezek **önmagukban nem mindig okoznak azonnali kárt**, de meghatározhatják, hogy a kezdeti futtatás egyszeri esemény marad-e, vagy olyan hozzáférési ponttá alakul, amely a támadó számára későbbi rendszerinterakciókat és vezérlési lehetőségeket teremt meg.



Inicializálás és környezetfelmérés

Inicializálás és környezetfüggő viselkedés

A trójai jellegű malware-ek működésében a futtatást követő **inicializálási szakasz** meghatározó szerepet tölt be. Ebben a fázisban a komponens felméri, hogy milyen környezetben fut, milyen jogosultságokkal rendelkezik, milyen rendszerjellemzők érhetők el, és vannak-e olyan körülmények, amelyek befolyásolhatják a további működését.

Ez a **környezetfüggő működés** több okból is fontos. Egyrészt lehetővé teszi, hogy a malware csak akkor aktiváljon bizonyos funkciókat, ha a célrendszer megfelelőnek tűnik. Másrészt segítheti az elemzői környezetek felismerését, például ha a malware sandboxra, virtuális gépre vagy hibakeresésre utaló jeleket keres. Ilyenkor előfordulhat, hogy a komponens nem hajtja végre a teljes működését, késlelteti azt, vagy látszólag inaktív marad.

A [Kaspersky Gootkit-elemzése](#) jól használható példa erre a technikai logikára. A cikk szerint a Gootkit loader komponense többféle virtuális géphez, sandboxhoz vagy debuggerhez kapcsolódó ellenőrzést is végez, vagyis a futtatás korai szakaszában **megpróbálja eldönteni, hogy valós célrendszeren vagy elemzői környezetben fut-e**. Ez a trójai működés egyik fontos előkészítő rétege: a komponens a környezet alapján módosíthatja a további viselkedését.

Az ilyen működés miatt ugyanaz a fájl eltérő környezetekben eltérő képet mutathat. Egy laborban vagy automatizált rendszerben csak kevés aktivitás látszik, míg más környezetben további komponensek

betöltése, kommunikációs kísérlet vagy konfigurációs művelet is megjelenhet. Ezért a trójai működés technikai megértéséhez a statikus vizsgálat mellett a futás közbeni viselkedés elemzése is szükséges.

Környezetellenőrzés és elemzői környezet felismerése

A Kaspersky Gootkitről szóló elemzése jól bemutatja a környezetfelmérés folyamatát. A vizsgált Gootkit loader több különböző módszert használ virtuális gépek, sandboxok és hibakereső környezetek felismerésére. A cikkben ezek az ellenőrzések már a loader futásának korai szakaszában megjelennek, és ha valamelyik virtualizációs vagy elemzői környezetre utaló feltétel teljesül, **a komponens nem a megszokott végrehajtási útvonalon halad tovább, hanem végtelen ciklusba kerül**. Ez a működés jól szemlélteti, hogy a loader már a futtatás korai szakaszában képes felismerni az elemzői környezetre utaló jeleket, és ezek alapján megváltoztatni a végrehajtás menetét.

A Gootkit esetében az **ellenőrzések több rétegre terjednek ki**. A loader vizsgálhatja a fájl nevét, a betöltött modulokat, a felhasználónevet, a számítógép nevét, bizonyos BIOS- és videó-BIOS értékeket, a processzor azonosítóját, a fizikai memória méretét, valamint a hálózati adapterhez kapcsolódó azonosítókól származtatható értékeket. Ezek az adatok önmagukban alapvető rendszerinformációk, azonban együttesen alkalmasak lehetnek arra, hogy a malware következtessen a futási környezet jellegére.

A cikkben szereplő technikai példák alapján az ilyen ellenőrzések többféle forrásból dolgozhatnak:

Vizsgált terület	Technikai példa	Elemzési jelentőség
Betöltött modulok	dbghelp.dll, sbiedll.dll keresése	Debuggerre vagy sandbox-környezetre utalhat
Felhasználónév	CurrentUser, Sandbox	Automatizált elemzői környezetre utaló név-minta lehet
Gépnév	SANDBOX, 7SILVIA	Sandbox vagy laborrendszer azonosítását segítheti
BIOS-értékek	QEMU, VBOX, BOCHS, VirtualBox	Virtualizált környezet jelenlétére utalhat
Processzorazonosító	például Xeon keresése	Bizonyos szerveres vagy virtualizált környezetek felismerésében használható
Memóriaméret	alacsony fizikai memória ellenőrzése	Sandboxokra jellemző erőforrásprofil felismerésére szolgálhat
MAC-címhez köthető értékek	virtualizációs gyártókra utaló azonosítók	VMware, VirtualBox, Parallels vagy más virtualizációs környezet felismerésére használható

A fenti példákból látható, hogy a környezetellenőrzés nem egyetlen technikára épül. A malware nem feltétlenül egy darab jel alapján dönt, hanem több kisebb indikátort vizsgálhat egymás után. Egyetlen felhasználónév vagy BIOS-érték önmagában még nem feltétlenül meghatározó, de ha több elemzői környezetre jellemző nyom egyszerre jelenik meg, az már befolyásolhatja a malware futási logikáját.

Droppers és loaderek szerepe

A dropper helye a hozzáférési láncban

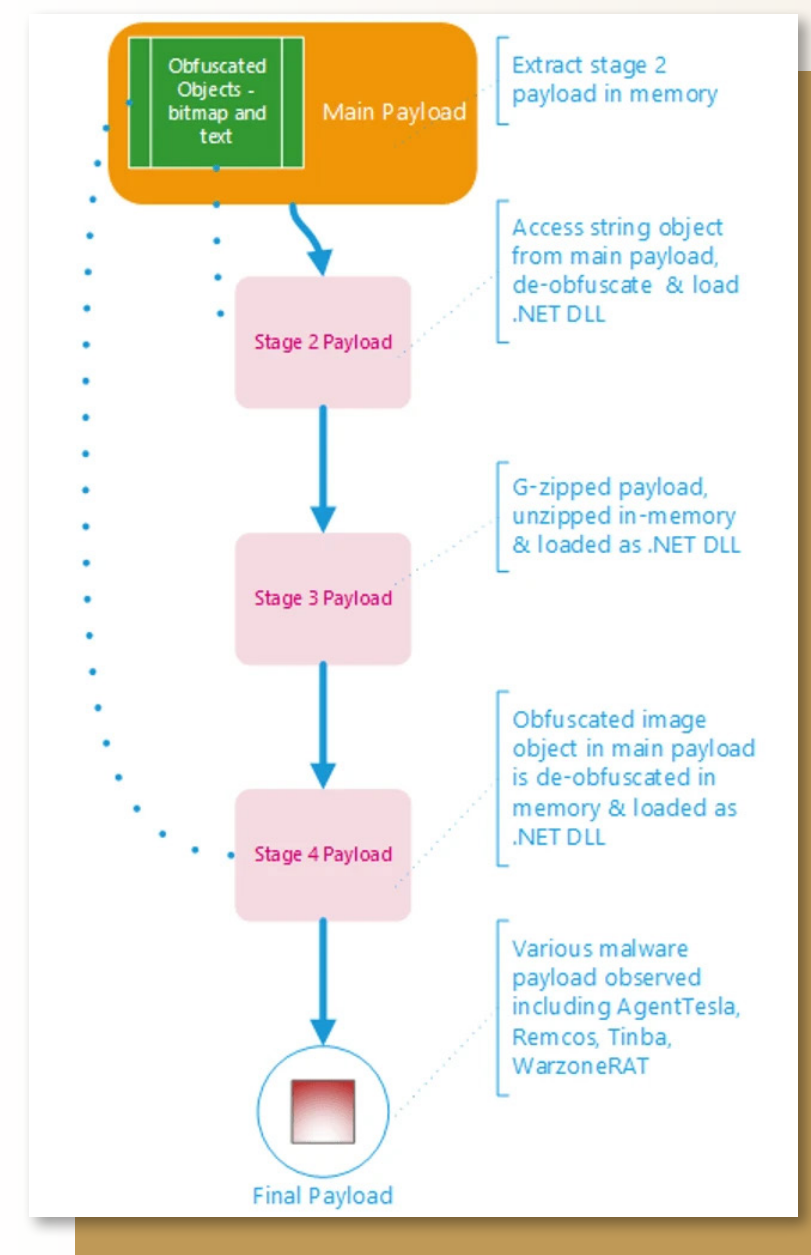
A dropper szerepét leginkább egy **előkészítő elemként** érdemes értelmezni. Elsődleges feladata, hogy a következő komponenst **eljuttassa arra a pontra, ahol az már futtathatóvá válik**. Ez lehet fájl kibontása, egy belső adatblokk visszafejtése, egy újabb modul memóriába töltése, vagy olyan technikai környezet kialakítása, amelyben a következő réteg már el tud indulni.

Ez azért lényeges, mert a kezdeti fájl vizsgálata sok esetben csak a lánc első elemét mutatja meg. A tényleges payload lehet elrejtve, tömörítve, titkosítva vagy több réteg mögé csomagolva, így a teljes működés csak futás közben válik láthatóvá.



Többlépcsős payload-kibontás

A dropperes működés egyik gyakori technikai formája a többlépcsős kibontás. Ilyenkor a kezdeti komponens először egy köztes réteget bont ki vagy tölt be, amely előkészíti a végső payload elindítását. Ez a köztes réteg újabb komponenst készíthet elő, így a teljes folyamat több egymásra épülő lépésből áll össze.



1. kép TicTacToe dropper extraction process

Forrás <https://www.fortinet.com/blog/threat-research/tictactoe-dropper>

A **Fortinet TicTacToe Dropperről** szóló elemzése jól szemlélteti ezt a működést. A leírás alapján **a dropper több DLL-réteget használt**, amelyeket futás közben, **egymást követően bontott ki és töltött be a memóriába**. Egy-egy réteg feladata a következő komponens dekódolása és előkészítése volt. Emiatt a teljes működés nem feltétlenül követhető pusztán a lemezen megjelenő fájlok alapján, mert a lánc egyes részei csak futás közben, a memóriában válnak láthatóvá.

Memóriában történő betöltés

A dropper elemzése során külön figyelmet érdemel, amikor a következő komponens **közvetlenül a folyamat memóriájába töltődik be, és ott aktiválódik**. Ilyenkor a hagyományos fájlalapú vizsgálat könnyen hiányos képet adhat, mert a végrehajtás egy része nem hagy egyértelmű fájlrendszeri nyomot.

A Fortinet elemzése alapján a TicTacToe Dropper több egymásra épülő rétegen keresztül készítette elő a későbbi komponenseket. Ezek nem minden esetben jelentek meg külön fájlként a rendszeren, hanem közvetlenül a memóriában kerültek futtatható állapotba. Emiatt az elemzés során a létrejövő fájlok mellett a futás közbeni betöltési folyamatokat és a memóriában megjelenő komponenseket is vizsgálni kell.

Dropper és loader kapcsolata

A **dropper és a loader** szerepe sokszor összekapcsolódik, ezért a gyakorlatban **nem mindig lehet őket elválasztani**. Általánosan a dropper inkább elhelyez vagy kibont egy következő komponenst, míg a loader a betöltésében, aktiválásában vagy további elemek lekérésében vállal szerepet. Egy kezdeti malware-komponens azonban mindkét

funkciót betöltheti: először kibont egy beágyazott réteget, majd az a réteg további payloadot készít elő vagy tölt be.

A hozzáférési malware-eknél ez azért fontos, mert a kezdeti futtatás sokszor még nem maga a tényleges hozzáférési funkció. A dropper inkább előkészíti a terepet: kibontja, elhelyezi vagy futtatható állapotba hozza azt a következő komponenst, amely később már a hozzáférés vagy a vezérlés kialakításában játszhat szerepet.



Backdoorok: rejtett hozzáférési pontok

A backdoor szerepe a hozzáférés megtartásában

A backdoor, vagyis „hátsó ajtó” lényege, hogy a **támadó** a kezdeti kompromittálás után is **vissza tudjon térni a rendszerbe**. A cél gyakran az, hogy a hozzáférési pont **ne okozzon feltűnő változást** a rendszer működésében, hogy lehetőséget adjon a későbbi támadói utasításokra.

A trójai és hozzáférési malware-ek láncában a backdoor már nem egyszerűen egy előkészítő elem. Inkább azt a pontot jelenti, ahol a kezdeti futtatásból **tartósabb rendszerjelenlét vagy távoli vezérlés lehetőség** alakulhat ki. A támadó számára ez azért értékes, mert nem kell minden alkalommal újra bejutási pontot keresnie: ha a backdoor működik, a rendszer később is elérhető maradhat.

Technikai oldalról ez **többféleképpen** is megvalósulhat. A fertőzött gép például időnként kapcsolatot kezdeményezhet egy külső vezérlő infrastruktúra felé, és ellenőrizheti, hogy érkezett-e új utasítás. Más esetben a hozzáférés csak bizonyos feltételek teljesülésekor aktiválódik. A **MITRE ATT&CK Traffic Signaling** módszere is ilyen működést ír le: egyes backdoorok nem folyamatosan tartanak fenn látható kapcsolatot, hanem csak egy előre meghatározott hálózati jelre aktiválnak rejtett funkciókat, például **port knocking** esetén egy adott portokra küldött kapcsolódási sorrend alapján. Ez jól szemlélteti, hogy a backdoor működését gyakran várakozási és megfigyelési szakaszok jellemzik, és csak meghatározott feltételek teljesülése esetén aktiválódik.

A visszatérés technikai feltételei

Egy backdoor jelentősége abban áll, hogy a kezdeti futtatást követően is képes visszatérni vagy újra aktiválódni. Ehhez technikai szinten mindig kell valamilyen kapaszkodó: egy fájl, egy modul, egy beállítás, egy indítási hivatkozás vagy egy olyan rendszermechanizmus, amely a komponenst ismét futásba hozza. Vagyis a **backdoor későbbi újraaktiválását** valamilyen, a rendszerben fennmaradó komponens vagy mechanizmus teszi lehetővé.

Ez a gyakorlatban többféle nyomot jelenthet. Előfordulhat, hogy a hozzá tartozó fájl egy kevésbé feltűnő könyvtárba kerül, legitimnek tűnő nevet kap, majd egy automatikus indítási pont hivatkozik rá. Más megoldásnál egy ütemezett feladat, szolgáltatás, felhasználói profilhoz kötött indítási beállítás vagy rendszerindításkor lefutó mechanizmus gondoskodik arról, hogy a komponens újra elinduljon.

Konkrét példaként említhető a Cisco Talos által elemzett **MagicRAT**, amely **több megmaradási pontot is használt**. A Talos írása szerint a minta ütemezett feladatot hozott létre, amely rendszerindításkor futtatta a malware komponensét, emellett pedig egy OneNote. Ink nevű hivatkozást is elhelyezett a Windows Startup mappájában. Egy ilyen hivatkozás tipikusan nem maga a kártékony komponens, hanem egy indítási elem: tartalmazhatja a célként megadott fájl elérési útját, az indításkor átadott paramétereket, a munkakönyvtárat, valamint az ikonhoz és megjelenítéshez kapcsolódó adatokat. A Windows ezeket az információkat használja fel ahhoz, hogy a parancsikon aktiválásakor a megfelelő célfájlt indítsa el.

A MagicRAT esetében ennek az volt a jelentősége, hogy a Startup mappába helyezett OneNote.lnk bejelentkezéskor **automatikus indítási pontként működhetett**. A névválasztás sem véletlen: a OneNote elnevezés legitim Microsoft-alkalmazásra emlékeztethet, ezért első ránézésre kevésbé feltűnő, mint egy ismeretlen vagy egyértelműen gyanús fájlnev. A backdoor tartós jelenlétét tehát a malware fájljának rendszeren való megtalálhatósága mellett olyan hivatkozások és ütemezett indítási pontok is biztosították, amelyek lehetővé tették a komponens ismételt futtatását.

Elemzői szempontból ez azért fontos, mert a backdoor **eltávolítása sem feltétlenül oldható meg egyetlen fájl törlésével**. Ha az indítási pont, a konfiguráció vagy a kapcsolódó betöltési folyamat megmarad, a hozzáférési komponens később ismét megjelenhet. Ezért a sikeres eltávolításhoz a teljes működési lánc feltárása szükséges, hogy a rendszerben ne maradjanak olyan elemek, amelyek lehetővé teszik a backdoor későbbi visszatérését.

Backdoorra utaló technikai nyomok

Egy backdoor felismerése nem mindig egyértelmű, mert **sok esetben nem látványos működéssel van jelen a rendszeren**. Éppen ezért nem elég csak azt figyelni, hogy fut-e egy gyanús folyamat. Azt is vizsgálni kell, hogy van-e olyan fájl, beállítás, indítási pont vagy hálózati kapcsolat, amely később újra aktiválhatja a hozzáférést.

Kézi vizsgálat során elsőként azokat a **Windows-felületeket és beállítási pontokat érdemes átnézni**, amelyek automatikus indításhoz kapcsolódhatnak. Ilyen például a **Feladatütemező**, ahol ellenőrizhetők az újonnan létrehozott vagy szokatlan időpontban futó ütemezett feladatok, illetve a **Szolgáltatások kezelőfelülete**, ahol a nem ismert, furcsa nevű vagy szokatlan elérési útról induló

szolgáltatások lehetnek érdekesek. Emellett érdemes megnézni a **Startup mappát** is, mivel az oda helyezett parancsikonok bejelentkezéskor automatikusan elindíthatnak programokat.

A futó folyamatok oldaláról érdemes megnézni, hogy egy háttérfolyamat honnan indult, milyen jogosultsággal fut, és milyen fájl tartozik hozzá. Erre használható például a **Process Explorer**, amely részletesebb képet ad a folyamatokról, mint az alap Feladatkezelő. A **Process Monitor** akkor lehet hasznos, ha azt kell követni, hogy egy gyanús folyamat milyen fájlokat, registry-bejegyzéseket vagy rendszererőforrásokat érint futás közben.

Naplózási oldalról a **Windows Eseménynapló** adhat fontos támpontot. Itt elsősorban a Windows-naplók alatt található **Biztonság, Rendszer és Alkalmazás naplókat** érdemes átnézni, mert ezekben megjelenhetnek gyanús bejelentkezések, új szolgáltatások, alkalmazáshibák vagy rendszerhez kapcsolódó események. Backdoor gyanú esetén különösen érdekes lehet, ha új szolgáltatás jött létre, ismeretlen folyamat indult el, vagy egy komponens rendszerindításhoz kötötten aktiválódott.

A backdoorra utaló jeleket **mindig együtt érdemes értékelni**. Egyetlen furcsa fájl, automatikus indítási bejegyzés vagy hálózati kapcsolat még nem bizonyít semmit. Ha viszont egy ismeretlen komponens automatikusan indul, szokatlan helyről fut, időnként külső címekkel kommunikál, és a naplókban ehhez kapcsolódó folyamat- vagy fájlváltozások is látszanak, akkor már olyan technikai lánc rajzolódik ki, amely rejtett hozzáférési pont jelenlétére utalhat.

RAT-ok: távoli vezérlés technikai modellje

A RAT szerepe a hozzáférési láncban

A RAT, vagyis **Remote Access Trojan** olyan malware-típus, amely a **kompromittált rendszert távolról vezérelhető eszközzé alakíthatja**. A backdoorhoz képest itt már nemcsak azon van a hangsúly, hogy a támadó később vissza tudjon térni a rendszerbe, hanem azon is, hogy a meglévő hozzáférést tényleges műveletekre tudja használni. Egyszerűbben megfogalmazva: a backdoor inkább hozzáférési útvonalat biztosít, a RAT pedig ezt a hozzáférést aktív vezérlési lehetőséggé bővíti.

Technikai szempontból egy RAT általában két oldalból áll: a fertőzött gépen futó komponensből és a vezérlőoldali infrastruktúrából. A fertőzött oldali komponens azonosíthatja a rendszert, állapotinformációkat küldhet, majd utasításokat fogadhat.

Fontos elkülöníteni a RAT-ot a legitim távoli hozzáférési eszközöktől. Utóbbiak önmagukban nem kártékonyak, de támadók visszaélhetnek velük. A RAT ezzel szemben **eleve kártékony komponens**, amely rejtett vagy megtévesztő módon biztosít távoli vezérlést. A **MITRE ATT&CK Remote Access Software** technikája ezért inkább összehasonlítási pontként használható, nem a RAT közvetlen megfelelőjeként.

A fertőzött oldali komponens működése

Egy RAT működése során a fertőzött gépen futó komponens általában **kapcsolatot tart egy vezérlőoldallal**, amelyen keresztül utasításokat fogadhat és adatokat továbbíthat. A futás korai szakaszában gyakran összegyűjt néhány alapvető

rendszeradatot, például a gép nevét, a felhasználónevet, az operációs rendszer verzióját, a hálózati környezetet vagy a folyamat azonosítóit. Ezek az adatok segítenek a vezérlőoldalon megkülönböztetni az egyes fertőzött rendszereket, és eldönteni, hogy az adott gépen milyen további műveletek lehetnek értelmezhetők.

A kapcsolat jellemzően **parancs-válasz logikára épül**. A fertőzött oldali komponens bejelentkezik a vezérlőoldalra, elküldi az állapotát, majd ellenőrzi, hogy tartozik-e hozzá új utasítás. Ha igen, végrehajtja a kapott műveletet, majd az eredményt visszaküldi. Ez lehet egyszerű rendszerinformáció-lekérés, fájlművelet, folyamatindítás vagy parancssori művelet. A technikai lényeg abban rejlik, hogy a fertőzött rendszer **távolról utasítható végponttá válik**, amelynek működését **a támadó a saját igényei szerint** irányíthatja.

A MITRE ATT&CK a távoli hozzáférési eszközöknél kiemeli, hogy ezek grafikus felületen, parancssori kapcsolaton, protokoll-alagúton vagy más távoli munkameneten keresztül is kapcsolatot teremthetnek két rendszer között. Ez jól szemlélteti, hogy a RAT működése különböző technikai megoldásokra épülhet, és **többféle távoli vezérlési formát** is magában foglalhat.

Távoli vezérlési képességek

A RAT-oknál a legfontosabb különbség a sima backdoorhoz képest az, hogy itt a **hozzáférés már aktívan használható**. A támadó nemcsak vissza tud térni a rendszerhez, hanem különböző műveleteket is indíthat rajta. A gyakori képességek közé tartozhat a rendszerinformációk lekérése, a fájlok listázása, fájlok feltöltése vagy letöltése, fájlok átnevezése, mozgatása vagy törlése, új folyamatok indítása, parancssori utasítások futtatása, valamint további komponensek előkészítése. Ezeket a képességeket ebben a fejezetben abból

a szempontból érdemes vizsgálni, hogy a RAT hogyan alakítja át a kompromittált gépet távolról kezelhető rendszerré. Konkrét példaként említhető a Cisco Talos által elemzett **MagicRAT**. A Talos leírása szerint a MagicRAT **távoli shell-hozzáférést biztosított**, amelyen keresztül tetszőleges parancsok végrehajtására volt lehetőség. Emellett **fájlkezelési műveleteket is támogatott**, például fájlok átnevezését, mozgatását és törlését. Ez a példa jól mutatja, hogy a RAT elsődleges szerepe a kompromittált rendszer távoli vezérlése és a különböző műveletek végrehajtásának biztosítása.

Komponensek kezelése és további műveletek előkészítése

A RAT másik fontos szerepe, hogy rajta keresztül a támadó további komponenseket kezelhet vagy készíthet elő. Ez túlmutat az egyszerű fájlműveleteken. Nemcsak arról van szó, hogy fájlokat tud listázni, mozgatni vagy törölni, hanem arról is, hogy új eszközöket tölthet le, meglévő komponenseket cserélhet, konfigurációkat módosíthat, vagy egy következő támadási lépéshez szükséges fájlokat juttathat a rendszerre.

Ez a működés azért lényeges, mert a RAT **rugalmasságot ad a támadónak**. Nem szükséges minden képességet már az első malware-komponensbe beépíteni. **Elég, ha a rendszer távolról vezérelhetővé válik, és a további funkciók csak később jelennek meg**. Így a RAT köztes vezérlési pontként működhet: fenntartja a kapcsolatot, fogadja az utasításokat, majd lehetőséget ad újabb komponensek aktiválására vagy a meglévő hozzáférés bővítésére.

Egyes RAT-ok megfigyelési vagy adatgyűjtési képességekkel is kiegészülhetnek, például billentyűleütések, képernyőhöz kapcsolódó adatok vagy felhasználói aktivitás rögzítésével. Ezek ebben a fejezetben

nem kerülnek részletesen ismertetésre, mert a keyloggerek és spyware-ek már inkább a célorientált malware-ek közé tartoznak. A lényeg, hogy a RAT a **távoli vezérlés mellett olyan műveletek előkészítésére is alkalmas lehet**, amelyek később más komponensekhez vagy további hozzáférési lépésekhez kapcsolódnak.

```
CT' ) function ( import , 0 , 1 ) Ma
get ( str( actions ) , 9sd , 1 ) t
all % dividedTarget . init ( action
3 , to return ] object . back ( acti
test ( action= 'SELECT' ) function (
select_all ( deviceDetection = to )
0 , 1 ) Remote Access Trojan array [
, 8 , 0 , 0.01 , 9 , 8 ] attack . te
, 3 , 1 , 7 ] array . select_all ( ri
t , 8 , return Math.max(def,nano) ] w
urn ] object . select_all ( action= 'S
return ] object . split ( burn || valu
return myObject ] object . init ( heig
```

Botnet kliensek és vezérlési logika

A botnet kliens szerepe a fertőzött rendszerben

A botnet kliens olyan fertőzött oldali komponens, amely a kompromittált gépet egy **nagyobb, támadó által irányított hálózat részévé teszi**. A működés középpontjában a nagyszámú fertőzött végpont **együttes és összehangolt vezérlése** áll.

A **fertőzött gép** ilyenkor „botként” működik: kapcsolatot tart a vezérlőoldallal, időnként bejelentkezik, fogadja a neki kiosztott parancsokat, majd végrehajtja azokat. Ez a kapcsolat teszi lehetővé, hogy az egyes fertőzött rendszerek egy központilag irányított hálózat részeként, **összehangoltan hajtsák végre** a kapott utasításokat.

Parancsok kiosztása és végrehajtása

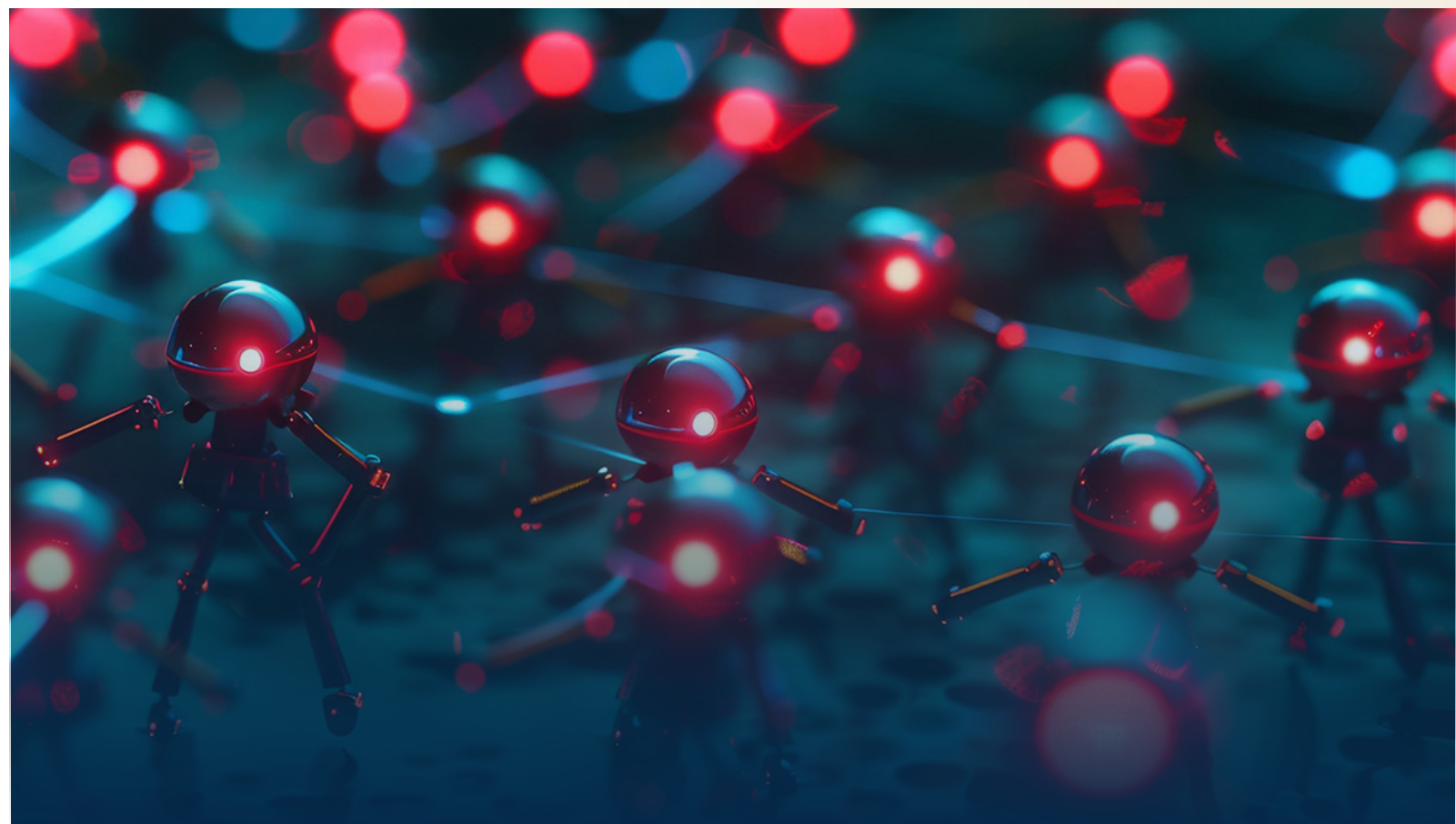
A botnet kliens egyik legfontosabb technikai szerepe, hogy **utasításokat tud fogadni a vezérlőoldaltól**. A kliens meghatározott időközönként bejelentkezhethet, lekérheti a neki szánt parancsot, végrehajthatja azt, majd visszaküldheti az eredményt vagy az állapotváltozást.

A kapott **utasítás többféle lehet**. A kliens kaphat parancsot rendszerinformációk lekérésére, fájlok kezelésére, új komponens letöltésére, konfiguráció módosítására vagy valamilyen hálózati művelet előkészítésére. A működés technikai lényege, hogy a fertőzött gép egy **központi vezérlési logika alapján hajtja végre** a számára meghatározott feladatokat. Botneteknél ez azért különösen fontos, mert ugyanaz az utasítás egyszerre sok fertőzött géphez is eljuthat. A támadó így nemcsak egyetlen rendszeren tud műveletet indítani, hanem nagyobb számú végpontot is összehangoltan használhat.

Mire használható egy botnet kliens?

A botnet kliens azért értékes a támadó számára, mert **sok fertőzött gépet lehet vele egyszerre irányítani**. A fertőzött rendszerek kaphatnak azonos parancsot, de akár csoportonként eltérő feladatot is. Például más utasítást kaphatnak az aktív gépek, másikat az újonnan csatlakozó kliensek, és megint mást azok a rendszerek, amelyek valamilyen szempontból értékesebbnek tűnnek.

Számos műveletet hajthat végre egy botnet kliens. Ezek közé tartozik például a hálózati forgalom generálása, spamküldés támogatása, további malware-komponensek terjesztése, proxyként való felhasználás, adatgyűjtés előkészítése vagy újabb eszközök letöltése. A felsorolt műveletek alapján jól látható, hogy a botnet kliens egy nagyobb támadói infrastruktúra részévé teheti a kompromittált eszközöket.



C2 kommunikáció és hozzáférésfenntartás

A C2 kapcsolat szerepe

A C2, vagyis command and control kapcsolat az a kommunikációs útvonal, amelyen keresztül **a fertőzött rendszer kapcsolatot tart a támadó vezérlő infrastruktúrájával**. Ez a vezérlési pont lehet egy C2 szerver, domain, IP-cím, webes végpont vagy ezek több elemből álló kombinációja. A lényeg, hogy a malware ezen keresztül tud utasításokat kapni, állapotot jelezni, konfigurációt frissíteni vagy további komponenseket letölteni.

Ez a kapcsolat teszi lehetővé, hogy az egyszeri fertőzés később is használható hozzáféréssé váljon. Egy trójai lefuthat a rendszeren, egy loader betölthet új komponenst, egy RAT távoli vezérlést adhat, de ezek tartós támadói értéke nagyrészt attól függ, hogy van-e működő kommunikációs csatorna a fertőzött gép és a vezérlőoldal között.

Kapcsolatfelvétel és bejelentkezés

A C2 kommunikáció általában úgy indul, hogy a fertőzött gépen futó komponens kifelé kezdeményez kapcsolatot. Ez azért gyakori, mert a belső hálózathoz kifelé induló webes forgalom sok környezetben engedélyezett, míg a kívülről befelé érkező kapcsolatokat a tűzfalak gyakrabban blokkolják. Emiatt a malware sok esetben **maga létesít kapcsolatot** a vezérlőoldallal, amelyen keresztül fogadhatja a további utasításokat.

A bejelentkezés során a komponens küldhet egyedi kliensazonosítót, gépnevet, felhasználónevet, operációs rendszer-verziót, jogosultsági szintet, malware-verziót vagy rövid állapotjelzést. Ezek alapján

a vezérlőoldal látja, hogy melyik fertőzött rendszer aktív, milyen környezetben fut, és milyen feladatot lehet neki kiosztani.

A gyakorlatban ez leggyakrabban **HTTP vagy HTTPS alapú kommunikációval** jelenik meg, mert ez könnyebben beleolvadhat a normál webes forgalomba. Előfordulhat **DNS-alapú C2** is, ahol a lekérdezett domainnév vagy a DNS-válasz hordozhat vezérlési információt. Régebbi botneteknél IRC-alapú vezérlés is jellemző volt, ahol a botok egy csatornához csatlakoztak, és ott kaptak parancsokat.

Parancslekérés és válaszlogika

A C2 kapcsolat központi része a parancslekérés. A fertőzött komponens **meghatározott időközönként bejelentkezhet a vezérlőoldalra**, és ellenőrizheti, hogy tartozik-e hozzá új utasítás. Ha nincs feladat, akkor visszatér várakozó állapotba, majd később újra próbálkozik.

Ha a kliens kap parancsot, akkor azt **helyben végrehajtja**, majd **visszaküldi az eredményt vagy az állapotváltozást**. A kapott utasítás lehet például rendszerinformációk lekérése, fájl letöltése, új komponens indítása, konfiguráció frissítése, parancssori művelet futtatása vagy egy meglévő funkció módosítása.

A C2 működés sokszor **tartalék megoldásokat is tartalmaz**. Ha az elsődleges C2 végpont nem érhető el, a malware próbálkozhat másik domainnel, IP-címmel, URL-listával vagy akár más kommunikációs protokollal. Fejlettebb esetben több lépcsőben épül fel a vezérlés: a kezdeti C2 kapcsolat csak egy újabb konfigurációt, következő szervercímet vagy letöltési helyet ad vissza. Ez segíti, hogy a támadó ne veszítse el azonnal a hozzáférést, ha az elsődleges vezérlési pont kiesik.

Titkosítás és a forgalom elrejtése

A C2 forgalom gyakran titkosított vagy nehezen olvasható formában jelenik meg. Webes C2 esetén ez sokszor HTTP vagy HTTPS kommunikáción keresztül történik: a fertőzött komponens kérést küld a vezérlőoldal felé, a szerver válaszában pedig érkezik az új utasítás. A parancs tehát nem feltétlenül külön, látványos csatornán jelenik meg, hanem egy **normál webes kérés-válasz folyamatba** ágyazva. Az utasítás lehet például a HTTP-válasz törzsében, egy kódolt adatmezőben vagy titkosított üzenetként, amelyet a malware feldolgoz, értelmez, majd végrehajt.

A titkosítás történhet egyszerűen HTTPS kapcsolaton keresztül, de előfordulhat az is, hogy a malware **saját módon kódolja vagy titkosítja az üzeneteket**. Ennek célja, hogy a hálózati forgalom tartalma ne legyen könnyen értelmezhető, és ne derüljön ki egyszerűen, milyen parancs vagy válasz halad a fertőzött gép és a vezérlőoldal között. A végrehajtás után a kliens egy újabb kérésben visszaküldheti az eredményt, a státuszt vagy az állapotváltozást.

C2-re utalhat például, ha egy nem böngésző jellegű folyamat rendszeresen HTTPS kapcsolatot nyit ismeretlen domain felé, majd röviddel később új fájl jön létre, új folyamat indul vagy registry-módosítás történik. DNS-alapú kommunikációnál gyanús lehet a szokatlanul hosszú vagy véletlenszerűnek tűnő domainnév, a sok sikertelen lekérdezés, illetve az, ha a DNS-forgalom időben összekapcsolható más gyanús rendszereseményekkel.

Tartós jelenlét és jogosultsági szint

Automatikus újraindítási pontok

A hozzáférési malware-eknél fontos kérdés, hogy a komponens csak **egyszer fut-e le, vagy később is újra tud indulni**. Ha egy trójai, backdoor, RAT vagy botnet kliens csak az első futtatás idejére aktív, akkor a támadó hozzáférése könnyebben megszakadhat. Ezért sok malware valamilyen **automatikus indítási ponthoz kapcsolódik**, hogy rendszerindításkor, felhasználói bejelentkezéskor vagy meghatározott időpontban ismét futásba kerüljön.

Windows környezetben ez többféleképpen megjelenhet. Gyakori példa lehet a „Run” vagy „RunOnce” registry bejegyzések használata, a Startup mappába helyezett parancsikon, egy ütemezett feladat létrehozása, vagy egy Windows szolgáltatásként beállított komponens. Ezek közös célja, hogy a malware a rendszer normál működéséhez kapcsolódva, **automatikusan újrainduljon, és biztosítsa a tartós jelenlétét**.

Jogosultsági környezet és futási szint

A malware működését nagyban meghatározza, hogy milyen jogosultsági szinten fut. Egy alacsonyabb jogosultságú komponens általában csak korlátozottan fér hozzá fájlokhoz, beállításokhoz, folyamatokhoz vagy védettebb rendszerterületekhez. Ha viszont **magasabb jogosultságot** szerez, akkor már **mélyebben be tud avatkozni** a rendszer működésébe: szolgáltatásokat hozhat létre, rendszerbeállításokat módosíthat, más folyamatokhoz férhet hozzá, vagy tartósabb indítási pontokat alakíthat ki.

Ezért veszélyes már önmagában az is, ha a **felhasználó elindít egy kártékony fájlt**. A program nem feltétlenül rendszerszintű jogosultsággal indul, de a **felhasználó nevében** már így is képes lehet fájlokat olvasni vagy módosítani, hálózati kapcsolatot nyitni, adatokat gyűjteni, C2 szerverrel kommunikálni, vagy felhasználói szintű indítási pontokat létrehozni. Ha a felhasználó például hozzáfér bizonyos dokumentumokhoz vagy hálózati megosztásokhoz, akkor a nevében futó malware is megpróbálhat ezekhez hozzáférni.

A malware többféleképpen próbálhat magasabb jogosultsági szinthez jutni. Az egyik legegyszerűbb módszer a **felhasználó megtévesztése**: a program úgy jelenhet meg, mintha telepítő, frissítés vagy jogosnak tűnő eszköz lenne, majd emelt jogosultságot kér (Windows UAC (User Account Control) jogosultságkérő ablaka). Ha a felhasználó ezt jóváhagyja, a folyamat már nagyobb mozgásteret kaphat. Ilyenkor nem feltétlenül technikai sérülékenység történik, hanem a felhasználói jóváhagyás miatt kerül a program kedvezőbb futási helyzetbe.

Más esetben a malware **jogosultságemelési vagy UAC-megkerülési technikát** próbálhat használni. Ilyenkor a cél az, hogy a kezdetben alacsonyabb jogosultsággal futó komponens magasabb szintű műveletet tudjon elindítani, lehetőleg anélkül, hogy a felhasználó számára egyértelmű jóváhagyási helyzet jelenne meg. Ez nem feltétlenül úgy történik, hogy a malware közvetlenül saját magát emeli magasabb jogosultsági szintre; gyakran inkább egy legitim, emelt szinten futó Windows-mechanizmus működését próbálja eltéríteni.

Az Elastic Security Labs UAC-megkerülésekről szóló elemzése több ilyen mintát is bemutat. Ide tartozik például az automatikusan emelkedő Windows-programok végrehajtási útjának eltérítése, a registry-beállítások manipulálása, a DLL-betöltés eltérítése, emelt

jogosultságú COM-interfészek használata, ütemezett rendszerfeladatok kihasználása, illetve a Windows-könyvtárakhoz hasonló megtévesztő útvonalak alkalmazása. Ezekben a módszerekben a közös pont az, hogy a támadó egy olyan rendszerfunkciót próbál közvetítőként használni, amely bizonyos helyzetekben eleve magasabb jogosultsággal működik. Így a kártékony művelet a kezdeti, alacsonyabb jogosultságú folyamat helyett egy **magasabb jogosultsági szinttel** rendelkező végrehajtási környezetben valósulhat meg.

Védelmi oldalon éppen ezért fontosak azok a **funkciók, amelyek korlátozzák**, hogy egy **alkalmazás milyen fájlokhoz vagy mappákhoz férhet hozzá**. Windows környezetben ilyen például a Controlled Folder Access, magyarul a szabályozott mappahozzáférés, amely a fontosabb mappák védelmét segíti azzal, hogy az ismeretlen vagy nem engedélyezett alkalmazások nem módosíthatják szabadon a védett fájlokat. Ha egy program mégis hozzáférést igényel, azt külön engedélyezni kell. Ez jól mutatja, hogy a jogosultsági szint mellett a **hozzáférési szabályok is befolyásolják, mire képes ténylegesen egy futó komponens**.



Összegzés

A trójai és hozzáférési malware-ek jelentősége abban áll, hogy sok esetben nem azonnal látványos károkozással kezdik a működésüket. A fertőzés **első szakasza gyakran csendesebb**: egy fájl lefut, egy komponens elindul, kapcsolat épül ki, majd a háttérben kialakul egy olyan hozzáférési pont, amelyet a támadó később is használhat. Éppen ezért ezeknél a malware-eknél az aktuálisan végrehajtott műveletek mellett azt is fontos vizsgálni, hogy **milyen lehetőségeket teremtenek a későbbi tevékenységekhez**.

Az elemzés során láthatóvá vált, hogy a **trójai jellegű működés sokszor csak a kezdet**. A dropperek és loaderok további komponenseket készíthetnek elő, a backdoorok visszatérési lehetőséget biztosíthatnak, a RAT-ok távoli vezérlést adhatnak, a botnet kliensek pedig nagyobb, központilag irányított hálózat részévé tehetik a fertőzött rendszert. A C2 kommunikáció ezek között összekötő szerepet tölt be: ezen keresztül érkezhettek parancsok, visszajelzések, konfigurációk vagy újabb komponensek.

Ezek a működések azért veszélyesek, mert a **fertőzött rendszer kívülről akár továbbra is normálisnak tűnhet**. A felhasználó nem feltétlenül lát azonnali hibát vagy feltűnő változást, miközben a háttérben már kialakulhatott egy később is használható támadói hozzáférés. Ez különösen fontossá teszi a folyamatok, indítási pontok, jogosultsági szintek és hálózati kapcsolatok együttes vizsgálatát.

Összességében a hozzáférési malware-ek nem önálló, elszigetelt technikai elemekként értelmezhetők, hanem egy **támadási lánc részeiként**. A valódi kockázat gyakran a kezdeti futtatást követően bontakozik ki, amikor a támadó tartós jelenlétet, vezérlési lehetőséget

és további mozgásteret alakít ki. Ezért ezek elemzésekor a fertőzés tényén túl azt is vizsgálni kell, hogy **milyen hozzáférés jött létre**, és az **ennyi ideig maradhat észrevétlen** a rendszerben.



Források

Securelist by Kaspersky (**Anton Kuzmenko**) <https://securelist.com/gootkit-the-cautious-trojan/102731>
(Hozzáférés dátuma: 08 2026)

Fortinet (**Amey Gat és Mark Robson**) <https://www.fortinet.com/blog/threat-research/tictactoe-dropper>
(Hozzáférés dátuma: 08 2026)

MITRE ATT&CK (**Josh Day, Gigamon; Tony Lee**) <https://attack.mitre.org/techniques/T1205/>
(Hozzáférés dátuma: 08 2026)

Cisco Talos Intelligence Group (**Asheer Malhotra, Vitor Ventura, Jungsoo An**) <https://blog.talosintelligence.com/lazarus-magicroat/> (Hozzáférés dátuma: 08 2026)

Windows Forum <https://windowsforum.com/news/windows-threat-hunting-with-sysinternals-process-explorer-tcpview-autoruns-procmon-sysmon.399412/>
(Hozzáférés dátuma: 08 2026)

MITRE ATT&CK (**Dray Agha, @PurpleWolf, Huntress Labs; Matt**

Kelly, @breakersall; Zachary Stanford, @svch0st) <https://attack.mitre.org/techniques/T1219/>
(Hozzáférés dátuma: 08 2026)

Nemzetbiztonsági Szakszolgálat
Nemzeti Kiberbiztonsági Intézet
<https://nki.gov.hu/it-biztonsag/tudastar/robothalozat-botnet-2/>
(Hozzáférés dátuma: 08 2026)

Nemzetbiztonsági Szakszolgálat
Nemzeti Kiberbiztonsági Intézet
<https://nki.gov.hu/it-biztonsag/tudastar/mi-az-a-command-and-control-c2/>
(Hozzáférés dátuma: 08 2026)

Elastic (**Samir Bousseaden**) <https://www.elastic.co/security-labs/threat-command-exploring-windows-uac-bypasses-techniques-and-detection-strategies/>
(Hozzáférés dátuma: 08 2026)





NEMZETI
KIBERBIZTONSÁGI
INTÉZET



Kibertámadás!
podcast



Nemzetbiztonsági Szakszolgálat
Nemzeti Kiberbiztonsági Intézet



titkarsag@nki.gov.hu



nki.gov.hu



+36 (1) 325 7672

2026