OFFZONE 2026



Beyond Secure Desktop: Abusing and Hardening Windows Desktop Isolation

Igor Korkin

Independent researcher

WHO AM I



Research Profile

Igor Korkin, PhD

Academic & Practical Security Researcher

- Windows & Linux Security · OS Internals · AI-Driven Attacks
- Software Developer · Former Expert at Huawei
- Speaker at 30+ conferences · Black Hat x3 · IEEE S&P
- Author of 50+ papers · Huawei patent · Monograph 2026
- Scientific Advisor at MEPhI

AGENDA

- Hello keyspy!
- Fighting with keyloggers
- Windows Desktop
- CreateDesktop & access policy
- Keylogger vs protection apps:
  - Open-source: VeraCrypt, KeePass(XC), alphaDelta, DiskCryptor
  - Proprietary: Winlogon, Devolutions RDM
- DesktopRanger
- Conclusion & Discussion

PART I. WHAT ARE WE PROTECTING?

SENSITIVE INFORMATION

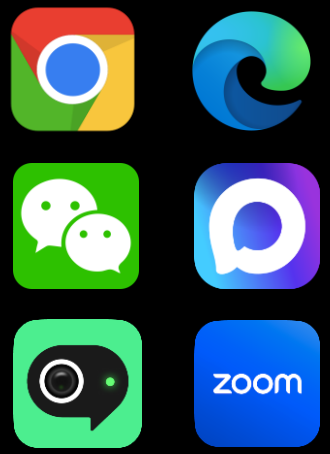
Web Browsing & Communication

Office & Documents

Passwords & Secrets

SENSITIVE INFORMATION

Web Browsing & Communication



Office & Documents



Passwords & Secrets

1Password

VeraCrypt

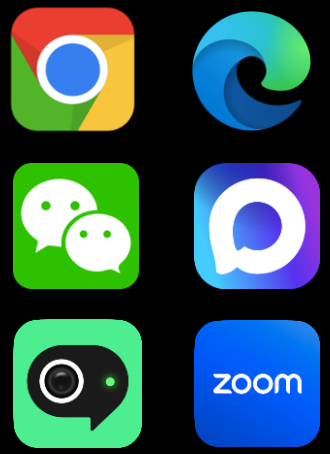
KeePass

Cryptocurrency wallets



SENSITIVE INFORMATION

Web Browsing & Communication



Office & Documents



Passwords & Secrets

1Password

VeraCrypt

KeePass

SENSITIVE INFORMATION

Web Browsing & Communication

Office & Documents

Passwords & Secrets

1Password

VeraCrypt

KeePass

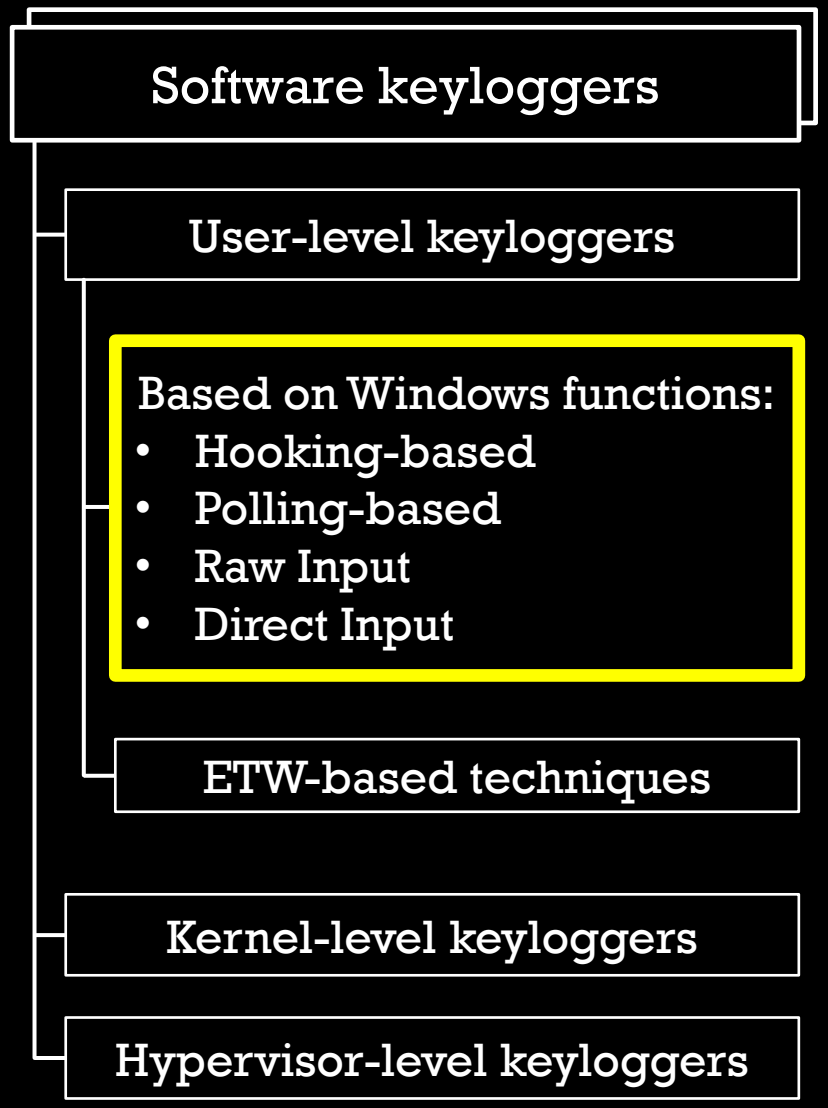


Keyloggers

KEYLOGGERS = WinAPI-based + high-privilege

- Software keyloggers
  - User-level keyloggers
    - Based on Windows functions:
      - Hooking-based
      - Polling-based
      - Raw Input
      - Direct Input
    - ETW-based techniques
  - Kernel-level keyloggers
  - Hypervisor-level keyloggers

KEYLOGGERS = WinAPI-based + high-privilege



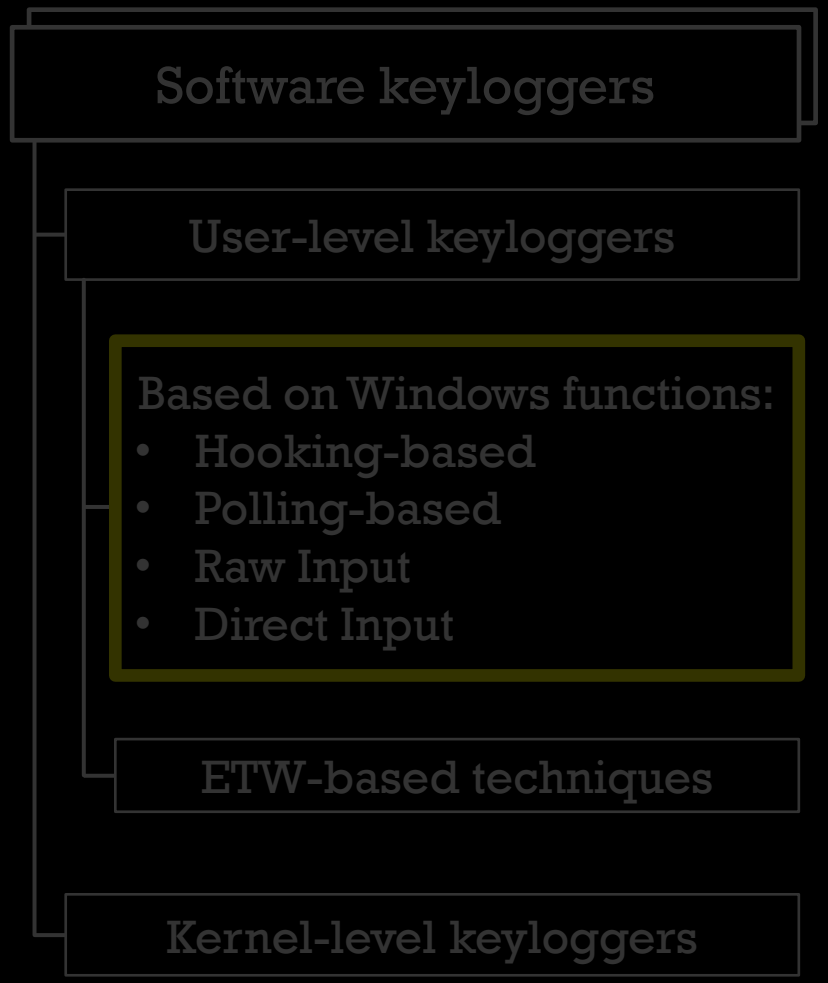
KEYLOGGERS = WinAPI-based + high-privilege

- Software keyloggers
  - User-level keyloggers
    - Based on Windows functions:
      - Hooking-based
      - Polling-based
      - Raw Input
      - Direct Input
    - ETW-based techniques
  - Kernel-level keyloggers
  - Hypervisor-level keyloggers

1 – Observed WinAPI-based keylogging techniques

WinAPI functions	Malware\Keyspy
SetWindowsHookEx	Snake
	Gurcu
GetAsyncKeyState	WarzoneRAT
	KLogExe
RegisterRawInputDevices GetRawInputData	Rhadamanthys
	RAT (3.3a), APT 31
DirectInput8Create GetDeviceState	EggStreme
	FusionCore

KEYLOGGERS = WinAPI-based + high-privilege



1 – WinAPI keylogging is still alive

WinAPI functions	Malware\Keyspy	Year
SetWindowsHookEx	Snake	2025
	Gurcu	2025
GetAsyncKeyState	WarzoneRAT	2025
	KLogExe	2024
RegisterRawInputDevices GetRawInputData	Rhadamanthys	2025
	RAT (3.3a), APT 31	2023
DirectInput8Create GetDeviceState	EggStreme	2025
	FusionCore	2023

KEYLOGGERS = WinAPI-based + high-privilege

- Software keyloggers
  - User-level keyloggers
    - Based on Windows functions:
      - Hooking-based
      - Polling-based
      - Raw Input
      - Direct Input
    - ETW-based techniques
  - Kernel-level keyloggers
  - Hypervisor-level keyloggers

1 – WinAPI keylogging is still alive

WinAPI functions	Malware\Keyspy	Year
SetWindowsHookEx	Snake	2025
	Gurcu	2025
GetAsyncKeyState	WarzoneRAT	2025
	KLogExe	2024
RegisterRawInputDevices GetRawInputData	Rhadamanthys	2025
	RAT (3.3a), APT 31	2023
DirectInput8Create GetDeviceState	EggStreme	2025
	FusionCore	2023

2 – High-privileged attackers

- MATE (Man-At-The-End) — remains in user mode, but controls the endpoint environment.
- CheckMATE 2025 @ ACM CCS

THREAT MODEL: USER-MODE KEYSPY WITH HIGH PRIVILEGES

Considered attacker:

✓ Documented WinAPI functions

✓ Privileges: User, Admin, SYSTEM*

Out of scope:

✗ SD & token manipulation

✗ memory forensics

✗ code injection



Experimental Keylogger

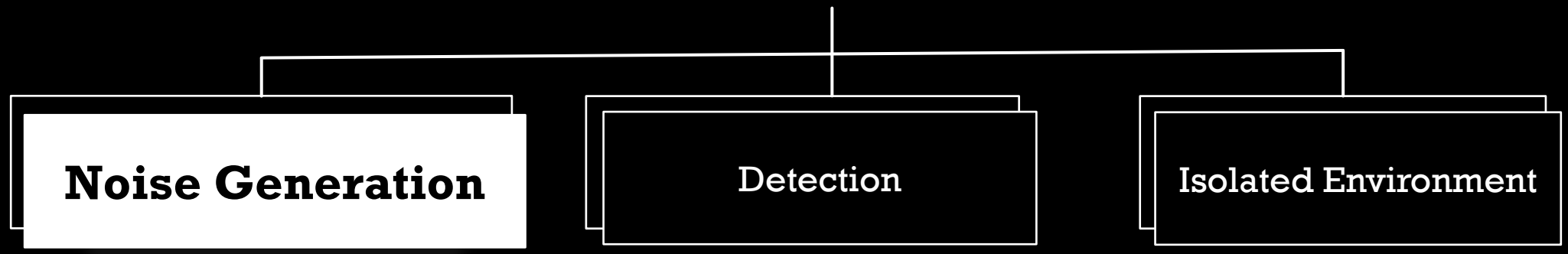
1. Implements 4 key-intercepting techniques:
   1. Hooks
   2. Raw Input
   3. DirectInput
   4. Polling
2. Monitors Desktop objects
3. Runs with different privileges
4. Logs: statuses, Desktop SD, keystrokes

- Source code:
  WinAPI + WIL / C++26 / ~3000 SLOC

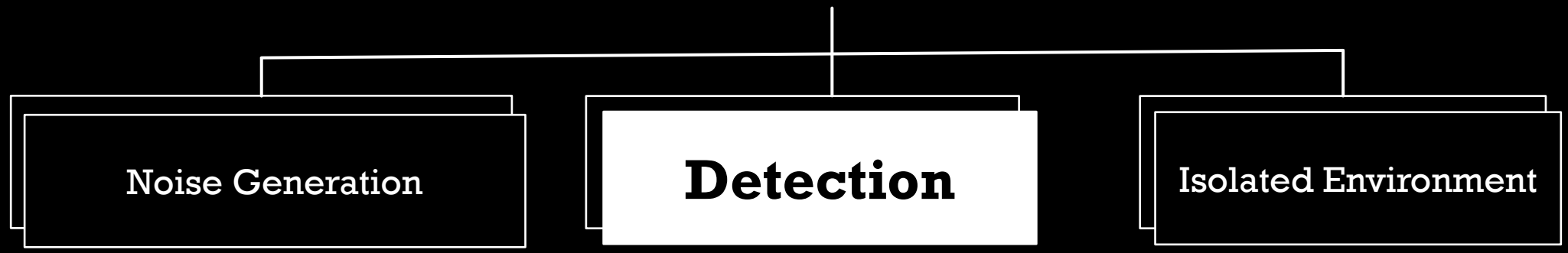
* SYSTEM is not full OS compromise. It is the most complicated user-mode scenario.

PART II. HOW DO WE FIGHT KEYLOGGERS?

FIGHTING KEYLOGGERS

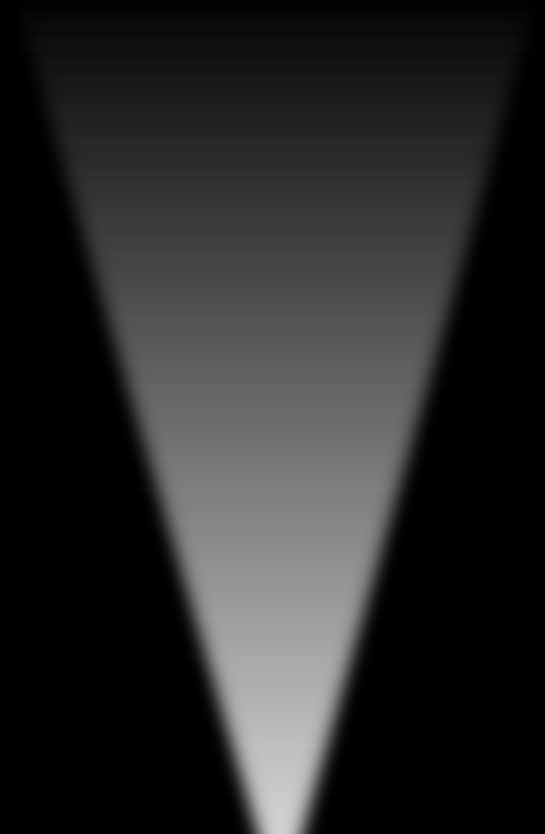


FIGHTING KEYLOGGERS



FIGHTING KEYLOGGERS

- Noise Generation
- Detection
- **Isolated Environment**



FIGHTING KEYLOGGERS

- Noise Generation
- Detection
- Isolated Environment
  - **Windows Desktop Technology**

Password Managers:


KeePass
1Password
VeraCrypt

Popular Password managers use Windows Desktop Technology for sensitive user input

FIGHTING KEYLOGGERS

Noise Generation

Detection

Isolated Environment

Windows Desktop Technology



Attackers can bypass protection technologies

FIGHTING KEYLOGGERS

Noise Generation

Detection

Isolated Environment

Windows Desktop Technology



FIGHTING KEYLOGGERS

Noise Generation

Detection

Isolated Environment



Windows Desktop Technology

Desktop Ranger



PART III. WINDOWS DESKTOP 101

A DESKTOP IS NOT A VIRTUAL DESKTOP

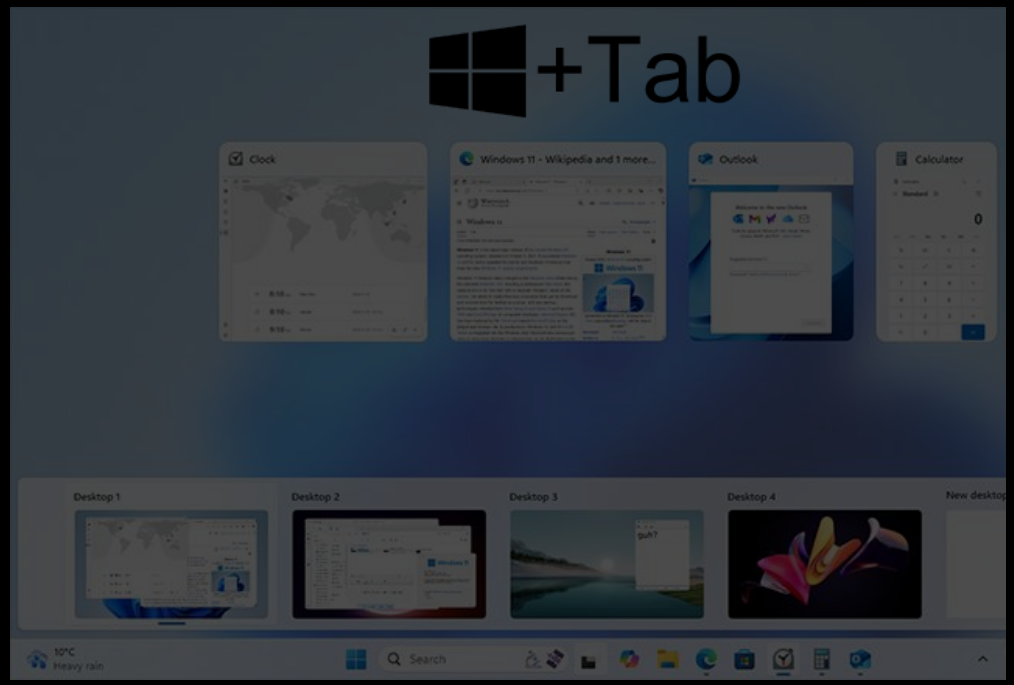
Windows 11 Task View



- a built-in workspace manager that can help to organize windows across multiple virtual desktops.
- not Win32 Desktop security objects.

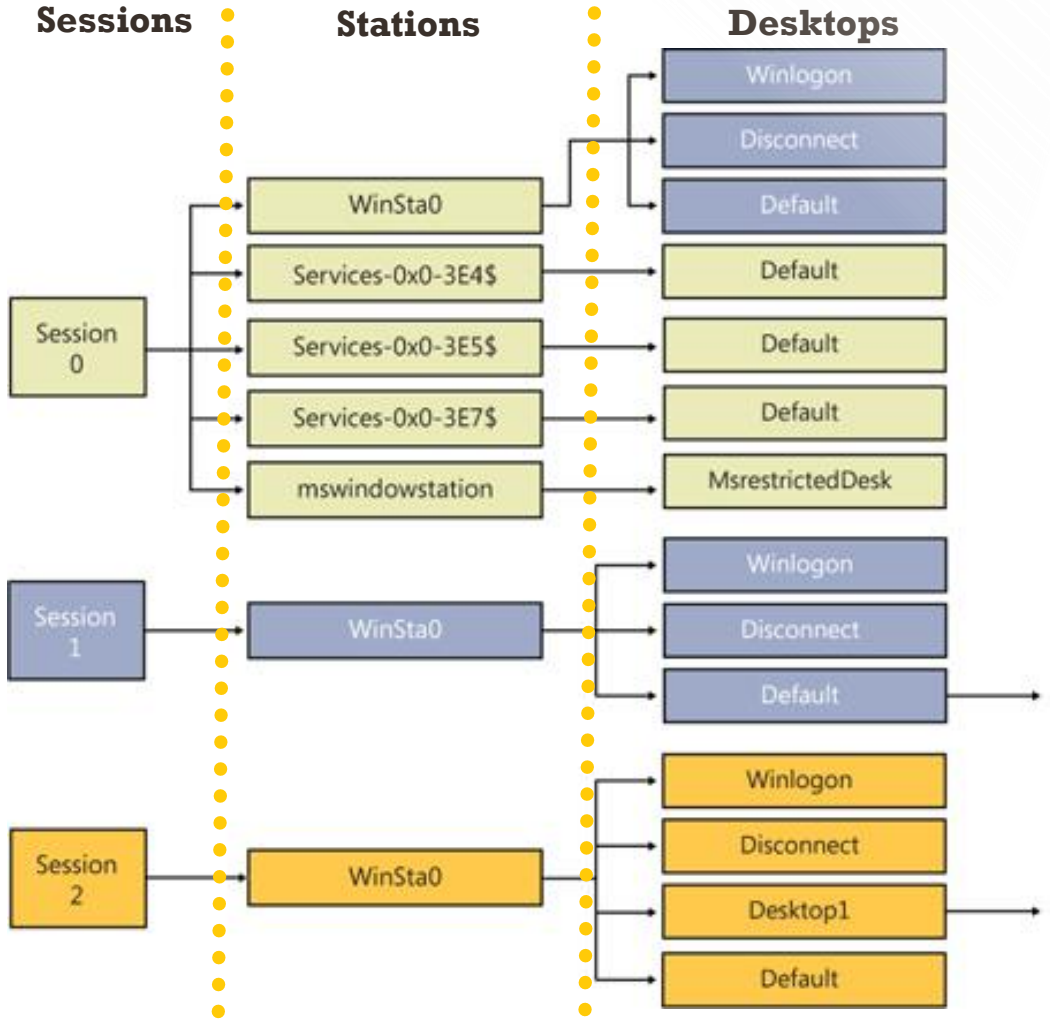
A DESKTOP IS NOT A VIRTUAL DESKTOP

Windows 11 Task View



- a built-in workspace manager that can help to organize windows across multiple virtual desktops.
- not Win32 Desktop security objects.

Windows Sessions, Stations, and Desktops



LOGIN → WINDOW STATION → WINDOWS DESKTOP



Window Station



LOGIN → WINDOW STATION → WINDOWS DESKTOP



Window Station



- Window station contains one or more desktop objects.
- A Desktop defines an interactive GUI/input context: windows, menus, hooks, message queues, and input-related mechanisms.

Window Station and Desktops are securable OS objects



Window Station



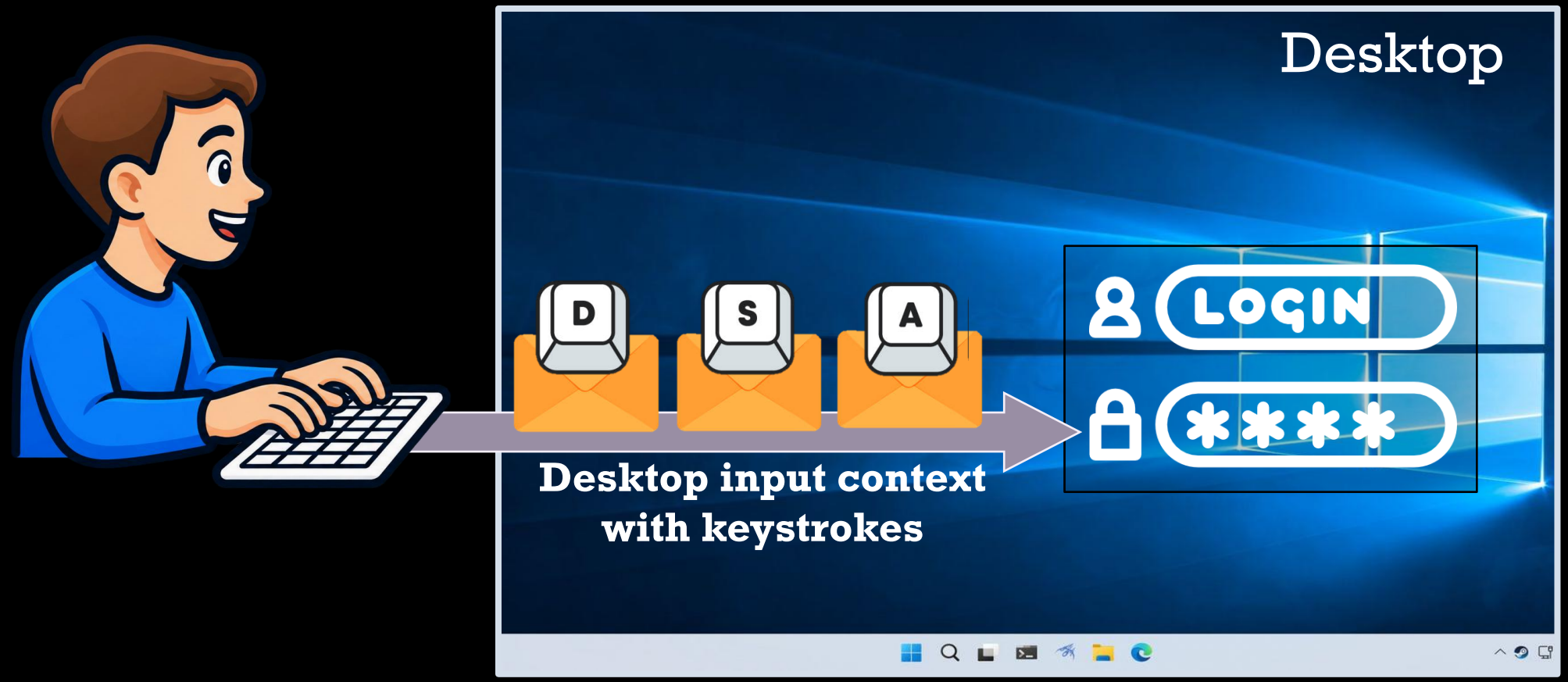
Station access rights:

- Create a desktop
- Enumerate desktops
- Read attributes
- etc

Desktop access rights:

- Create windows
- Install hooks
- Enumerate/read objects
- Switch / read / write

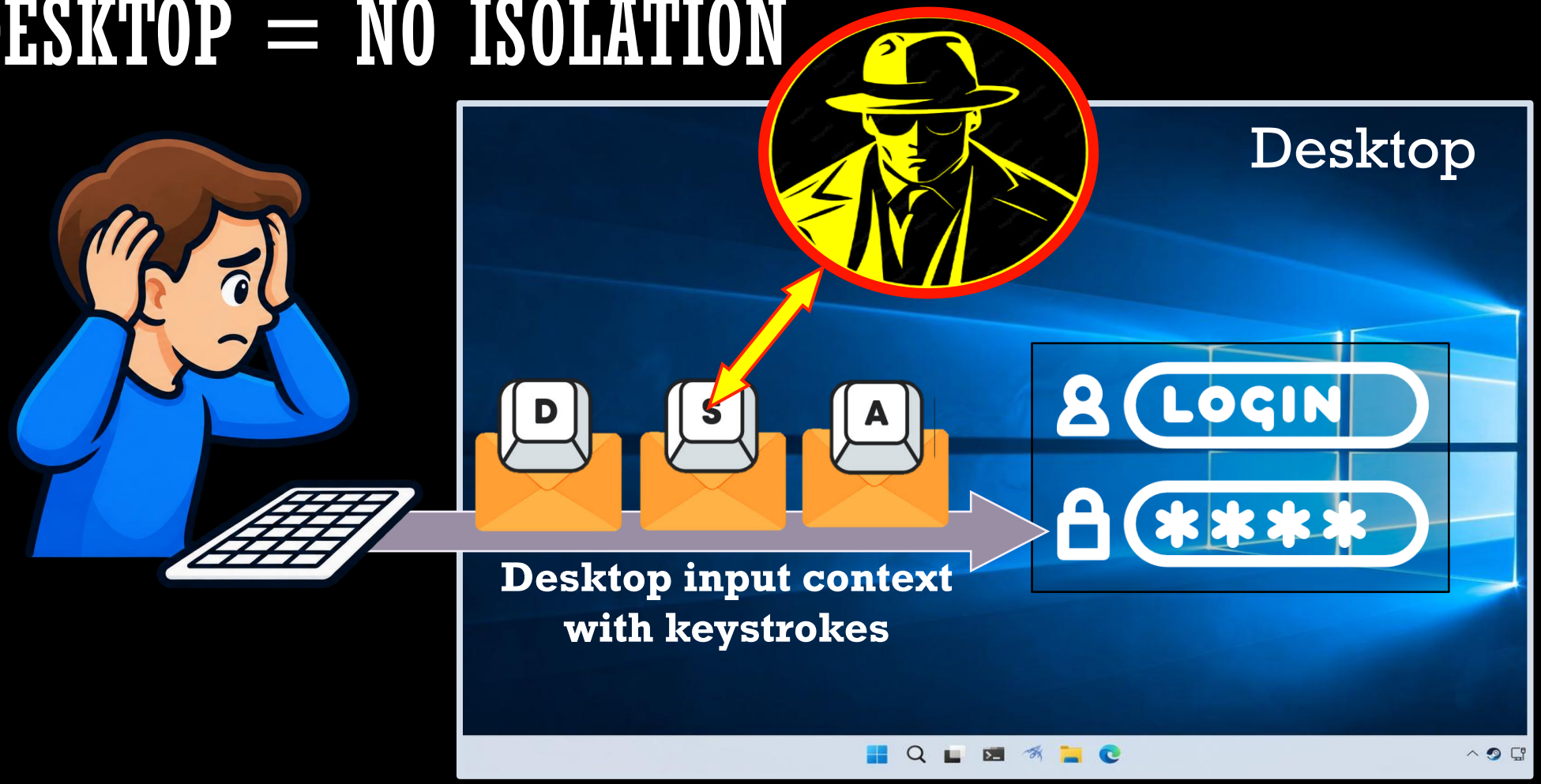
SAME DESKTOP = NO ISOLATION



A. Sensitive App is running on the Default Desktop

B. Keyspy is already inside the same Desktop context. Message hooks, polling, Raw Input, and DirectInput become relevant attack surfaces.

SAME DESKTOP = NO ISOLATION



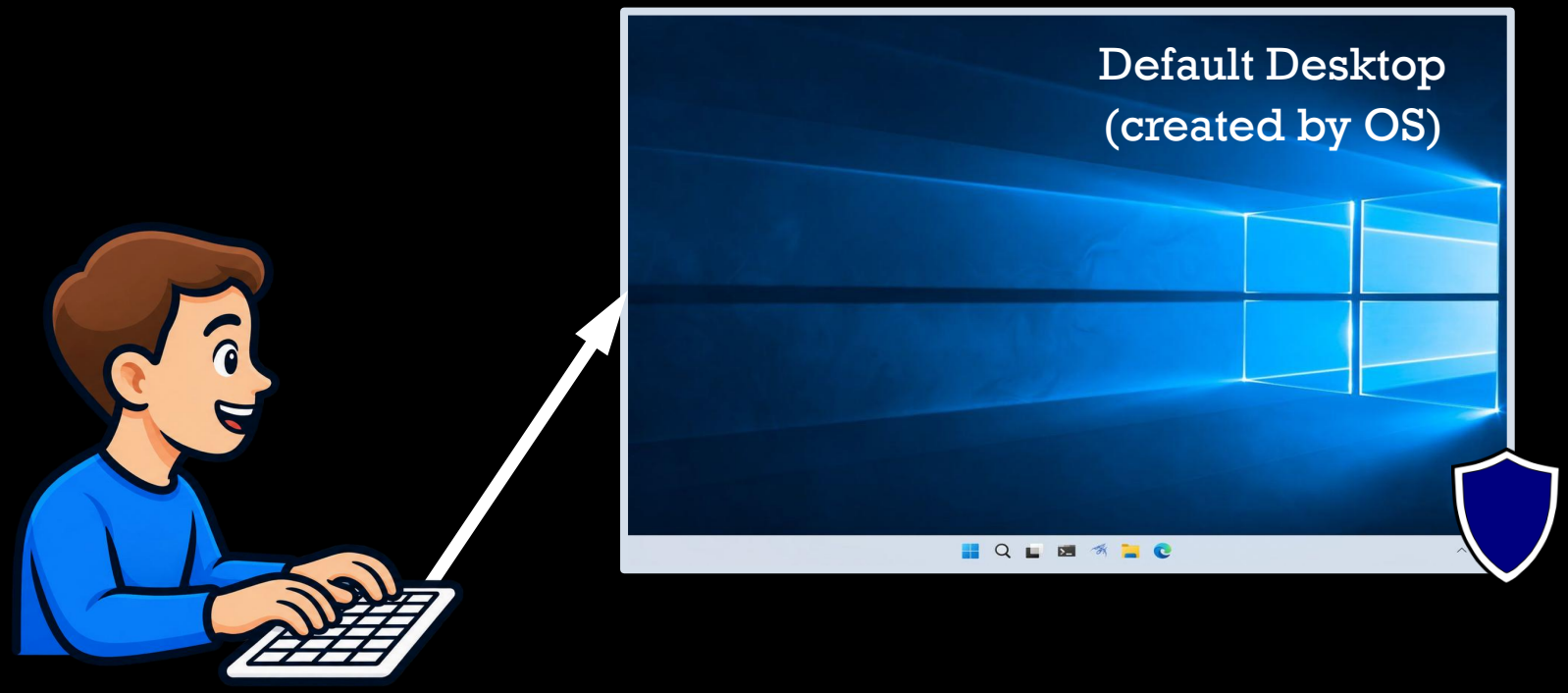
A. Sensitive App is running on the Default Desktop

B. Keyspy is already inside the same Desktop context. Message hooks, polling, Raw Input, and DirectInput become relevant attack surfaces.



How do we isolate the interactive input context?

LET'S CREATE A SECOND DESKTOP



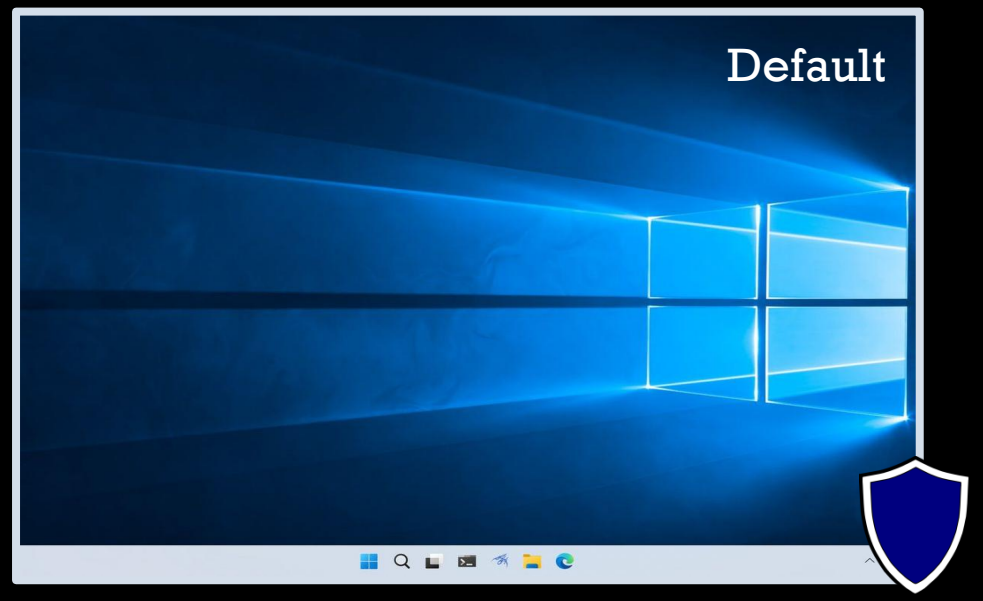
LET'S CREATE A SECOND DESKTOP

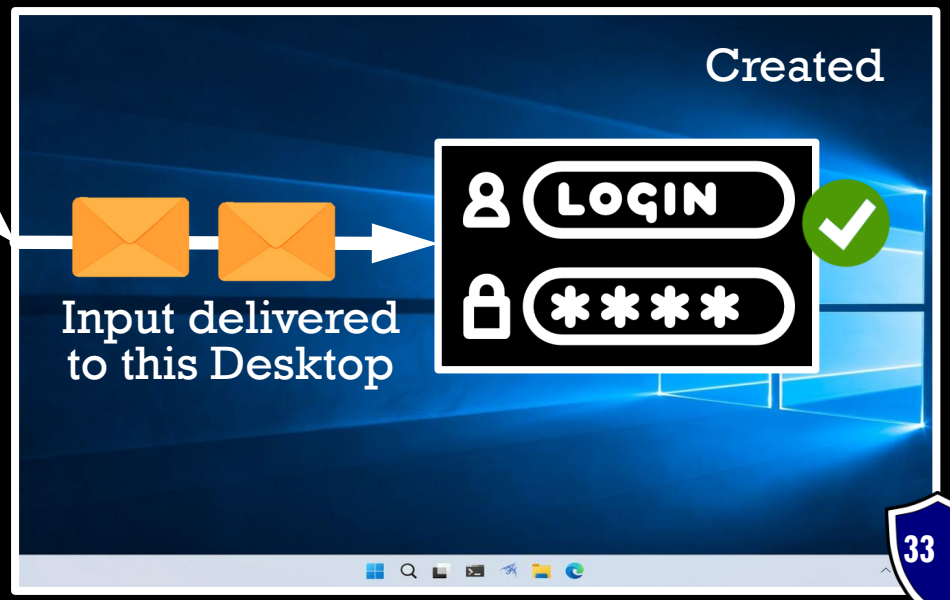


A. Password manager: creates a new desktop and relaunches its app on it

Password Managers:

KeePass 1Password VeraCrypt





LET'S CREATE A SECOND DESKTOP

A. A keyspy outside the new Desktop cannot observe that Desktop's input directly.

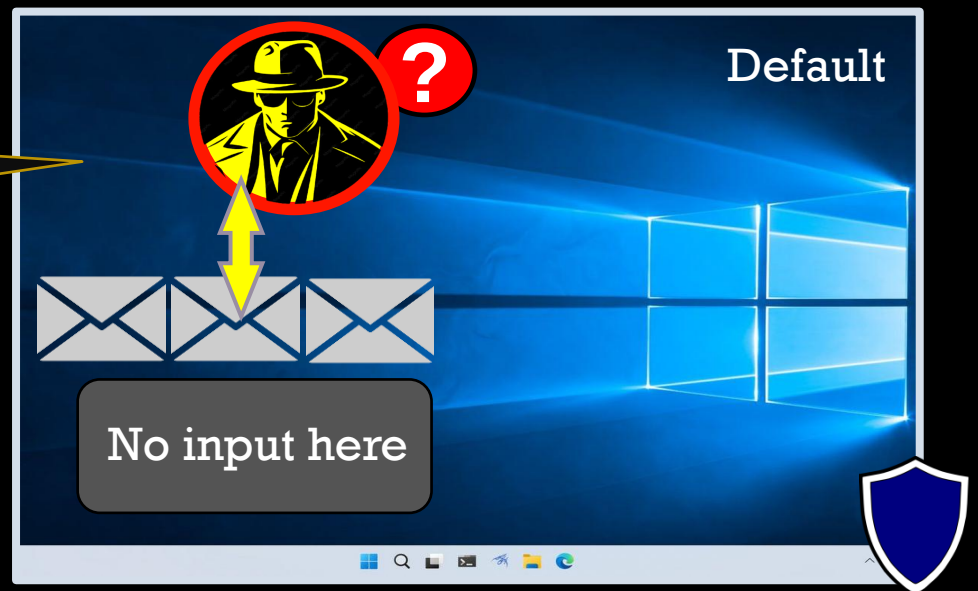


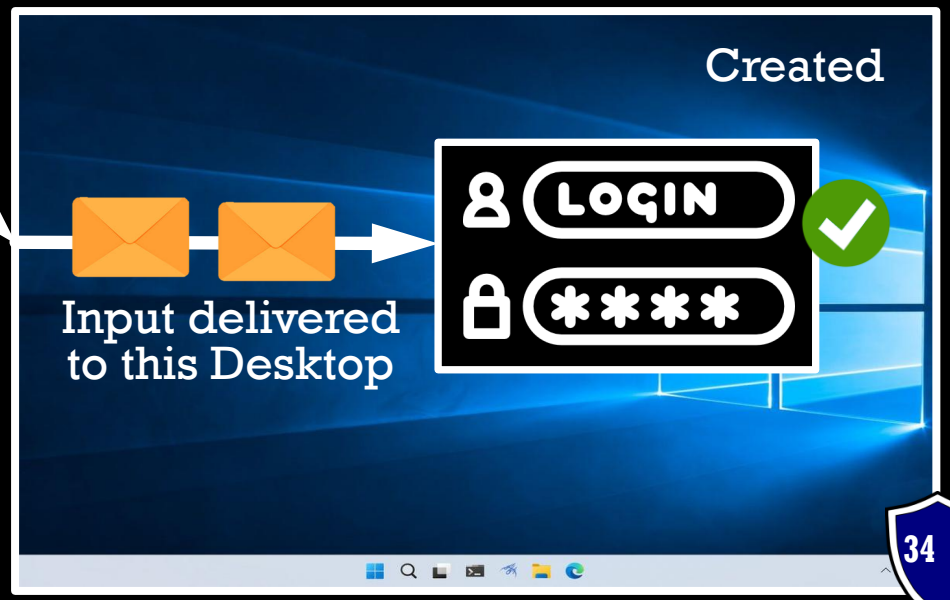
B. The new Desktop has a separate interactive GUI/input context

Password Managers:

KeePass 1Password VeraCrypt

No input from the protected Desktop here…







2nd Desktop protects user input only while the keyspy remains **outside** of the 2nd Desktop

Desktop Secure Object: Handle rights ≠ Object access policy

- CreateDesktop() creates a Desktop object and returns an HDESK

```
HDESK CreateDesktopW(
  [in]           LPCWSTR                lpszDesktop,      // Symbolic desktop name
                 LPCWSTR                lpszDevice,       // Reserved
                 DEVMODEW               *pDevmode,        // Reserved
  [in]           DWORD                  dwFlags,          // Reserved
  [in]           ACCESS_MASK            dwDesiredAccess,  // * Rights of the returned handle
  [in, optional] LPSECURITY_ATTRIBUTES  lpsa              // * Security Descriptor for
);                                                        //   the Desktop object
```

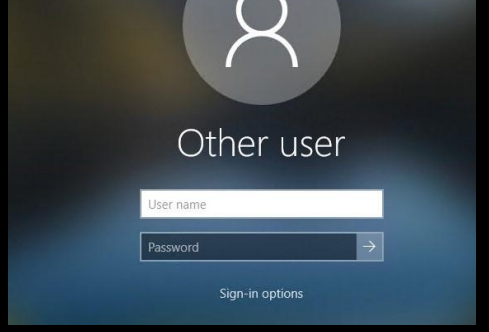
Parameter	What does it control
ACCESS_MASK dwDesiredAccess	Rights of the **returned Desktop handle**
SECURITY_ATTRIBUTES lpSecurityAttributes	**Security Descriptor** of the Desktop object

- DACL controls whether other processes can open the Desktop.
- lpSecurityAttributes = NULL → inherits Security Descriptor from the parent Window Station

ANALYSIS OF SECURE DESKTOP IMPLEMENTATIONS

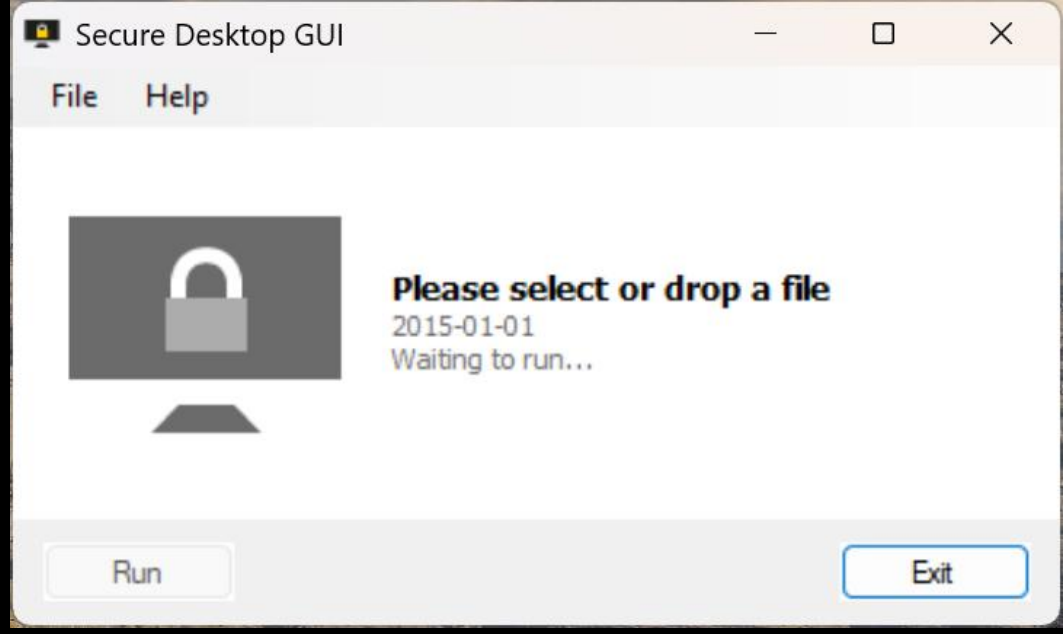
- Open-source software:
  1. alphaDelta Secure-Desktop
  2. DiskCryptor
  3. VeraCrypt
  4. KeePass
  5. KeePassXC

- Proprietary software:
  1. Devolutions Remote Desktop Manager (RDM)
  2. Winlogon Desktop



CASE 1: ALPHADELTA

Project	alphaDelta Secure-Desktop
src	SecureDesktop/Program.cs
Commit:	d5f2fc5 — "Added security measures"




```
IntPtr hNewDesktop = WinAPI.CreateDesktop(dname,
IntPtr.Zero, IntPtr.Zero, 0, (uint)WinAPI.DESKTOP_ACCESS.CUSTOM_SECURE, IntPtr.Zero);
```

lpSecurityAttributes=NULL

→ Desktop inherits parent Window Station SD

CASE 2: VERACRYPT

Project	VeraCrypt
src:	VeraCrypt/src/Common/Dlgcode.c
commit	cdbe54e6 — "Implement Secure Desktop for password entry"




```
hSecureDesk = CreateDesktop (pParam->szDesktopName, NULL, NULL, 0, desktopAccess, NULL);
```

lpSecurityAttributes=NULL

→ The Desktop inherited SD

CASE 3: DISKCRYPTOR

Project	DiskCryptor
Author:	David Xanatos
src:	DCrypt/gui/secure_desktop.c
Commit:	6b7bb35 — "2.0"



```
/* Create new desktop */
hSecureDesk = CreateDesktopW(pParam->szDesktopName, NULL, NULL, 0, desktopAccess, NULL);
```

lpSecurityAttributes=NULL

→ The Desktop inherited SD

CASE 4: KEEPASS

Project	keepass
src:	KeePass/UI/ProtectedDialog.cs
official:	KeePass 2.61.1 Source Code Package



KeePass

```
\KeePass-2.61.1-Source\KeePass\UI\ProtectedDialog.cs |  |1251 |Ln 138/48

        IntPtr pNewDesktop = NativeMethods.CreateDesktop(strName,
                null, IntPtr.Zero, 0, deskFlags, IntPtr.Zero);
```

KeePass Awards:

- BSI (Germany)
- ANSSI (France)
- DINUM (France)







lpSecurityAttributes=NULL

→ The Desktop inherited SD

CASE 5: KEEPASSXC – NO SECURE DESKTOP

KeePassXC

Project	KeePassXC
src:	src/gui/DatabaseOpenWidget.cpp
release:	KeePassXC 2.7.12
issue	#3460 — "Support Windows Secure Desktop when entering master password" — OPEN

Awards:

- French National Cybersecurity Agency (ANSSI)
- French government cybersecurity service

CYBER MALVEILLANCE .GOUV.FR

Support Windows Secure Desktop when entering master password #3460

Open · Feature

ThatDeaf-ITGuy opened on Aug 21, 2019

Summary

KeePass 2.x has a feature whereby the application takes the user to a "secure desktop" space to enter the master password. Per their documentation, this theoretically makes it harder (but not impossible) for keyloggers to log a user's master password. Like KeePass 2.x, this should probably be an "opt-in" feature.

Secure Desktop is not supported yet . . .

KeePassXC: enter your passwords on the Default Desktop.

CASE 5: KEEPASSXC – NO SECURE DESKTOP

KeePassXC

Project	KeePassXC
src:	src/gui/DatabaseOpenWidget.cpp
release:	KeePassXC 2.7.12
issue	#3460 — "Support Windows Secure Desktop when entering master password" — OPEN

Awards:

- French National Cybersecurity Agency (ANSSI)
- French government cybersecurity service

CYBER MALVEILLANCE .GOUV.FR

Support Windows Secure Desktop when entering master

Daedalus007 on Dec 5, 2025

--Short Version--

Cybersecurity expert here. I'd like to chime in on the convo if it is all the same to anyone else.

I'd like to propose a monetary bounty to implement this as an incentive if the KeePassXC authors/contributors don't personally feel like implementing this. If anyone wants this to happen, let's crowdfund it and make it happen. $2k+ USD should be enough, hmm? Start throwing some sponsor money at it. Ko-Fi could also be a potential option.

7 years later – Issue: still **open**
Users: Pls implement Secure Desktop!

Me: that's a research opportunity 😳

If your enemy can predict your next move — don’t move.



ANALYSIS

Product	Separate Desktop	lpSecurityAttributes
KeePassXC	⛔	(n/a)
KeePass	☑	NULL
VeraCrypt / DiskCryptor		
alphaDelta Secure-Desktop		

lpSecurityAttributes = NULL

⬇

inherited SD

⬇

hook allowed

⬇

desktop can be infected



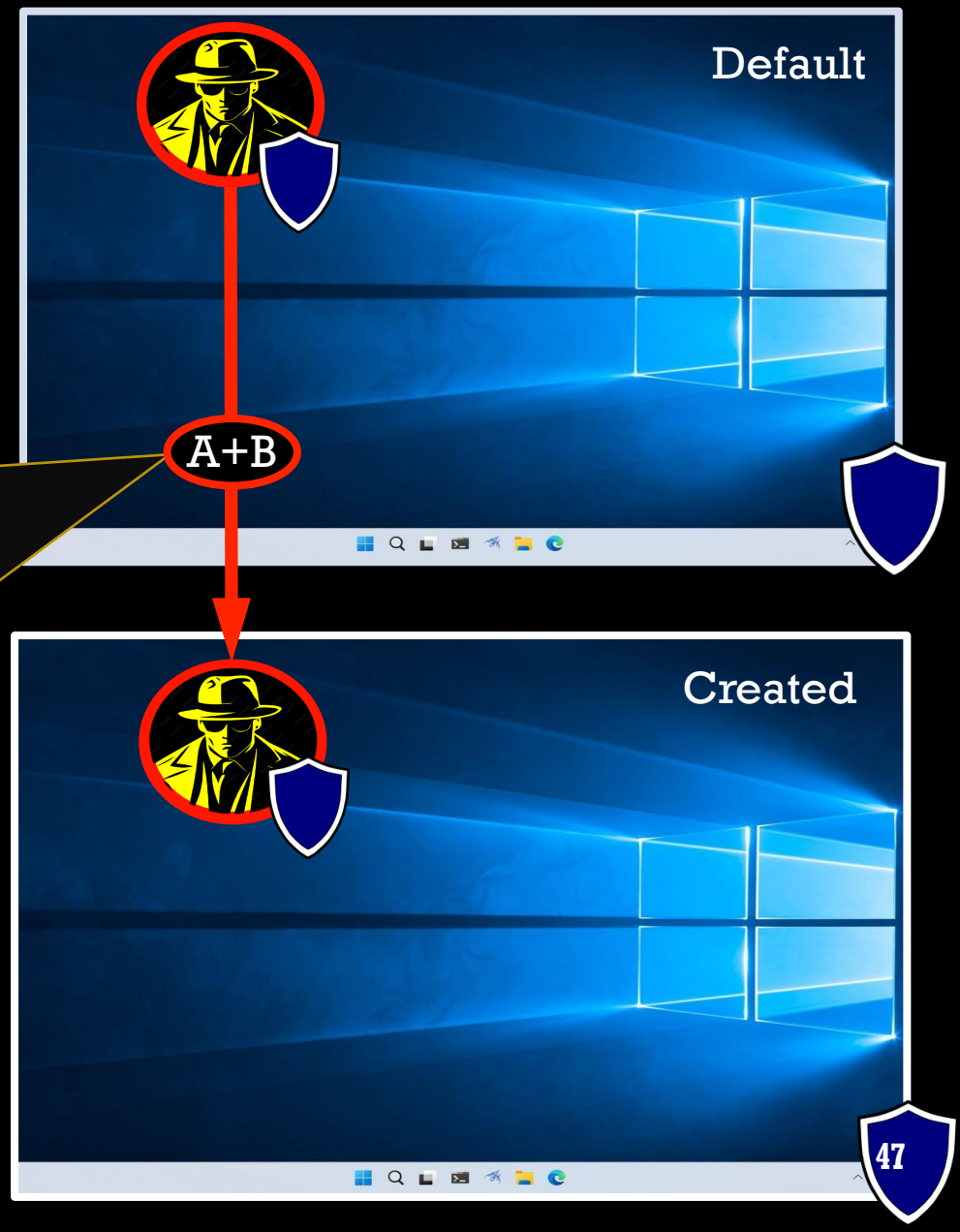
Can the 2nd Desktop be infected as well?

PRIOR ART: THIS ATTACK IDEA ALREADY EXISTS

If same-user attachment succeeds, the keyspy becomes part of the protected context:

Steps	WinAPI
A. Discover	EnumDesktops
B. Open target	OpenDesktop(name)
C. Launch process	CreateProcess + STARTUPINFO.lpDesktop
D. Install hook	SetWindowsHookEx

→ Now attacker can read all the keystrokes

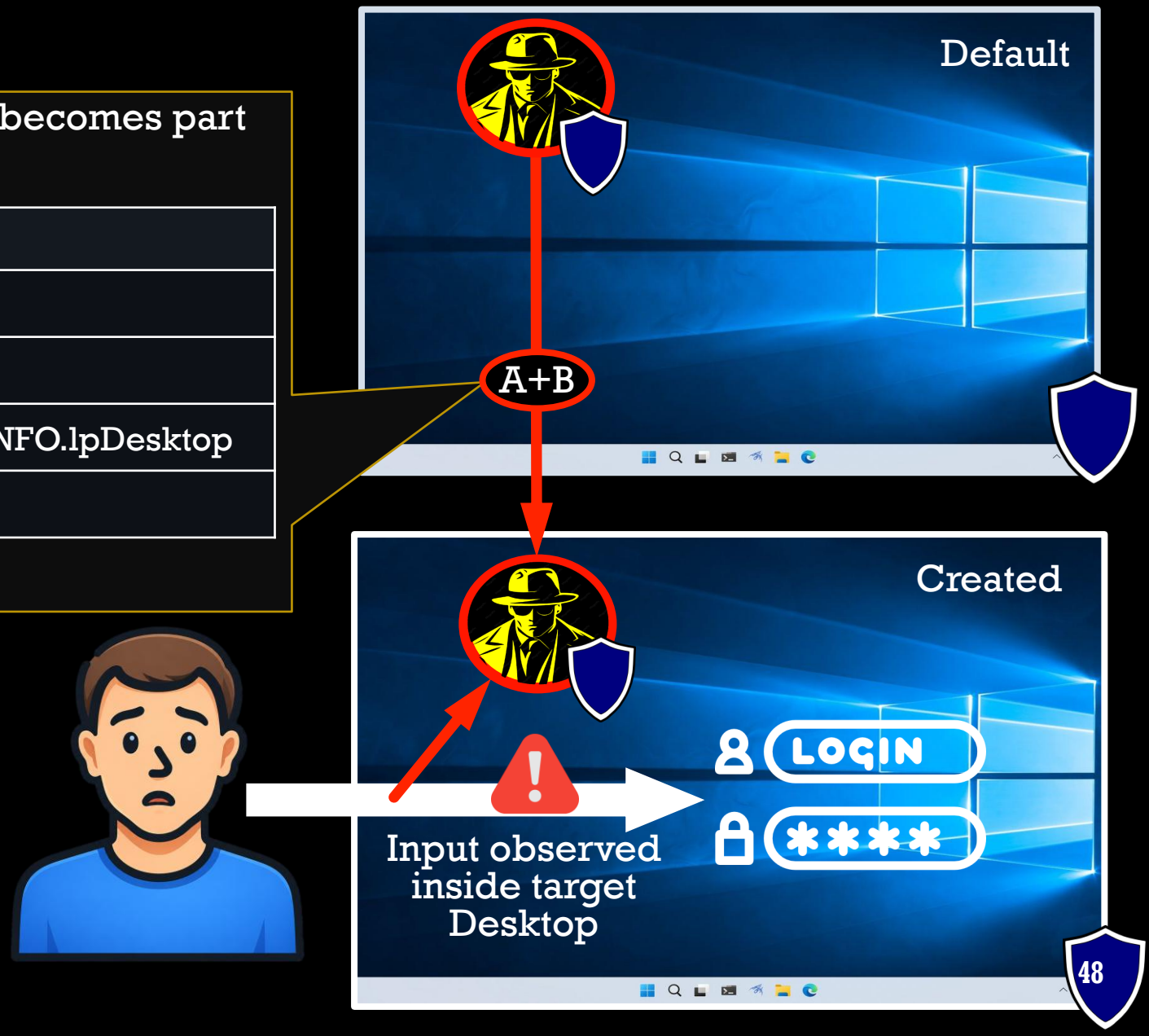


PRIOR ART: THIS ATTACK IDEA ALREADY EXISTS

If same-user attachment succeeds, the keyspy becomes part of the protected context:

Steps	WinAPI
A. Discover	EnumDesktops
B. Open target	OpenDesktop(name)
C. Launch process	CreateProcess + STARTUPINFO.lpDesktop
D. Install hook	SetWindowsHookEx

→ Now attacker can read all the keystrokes

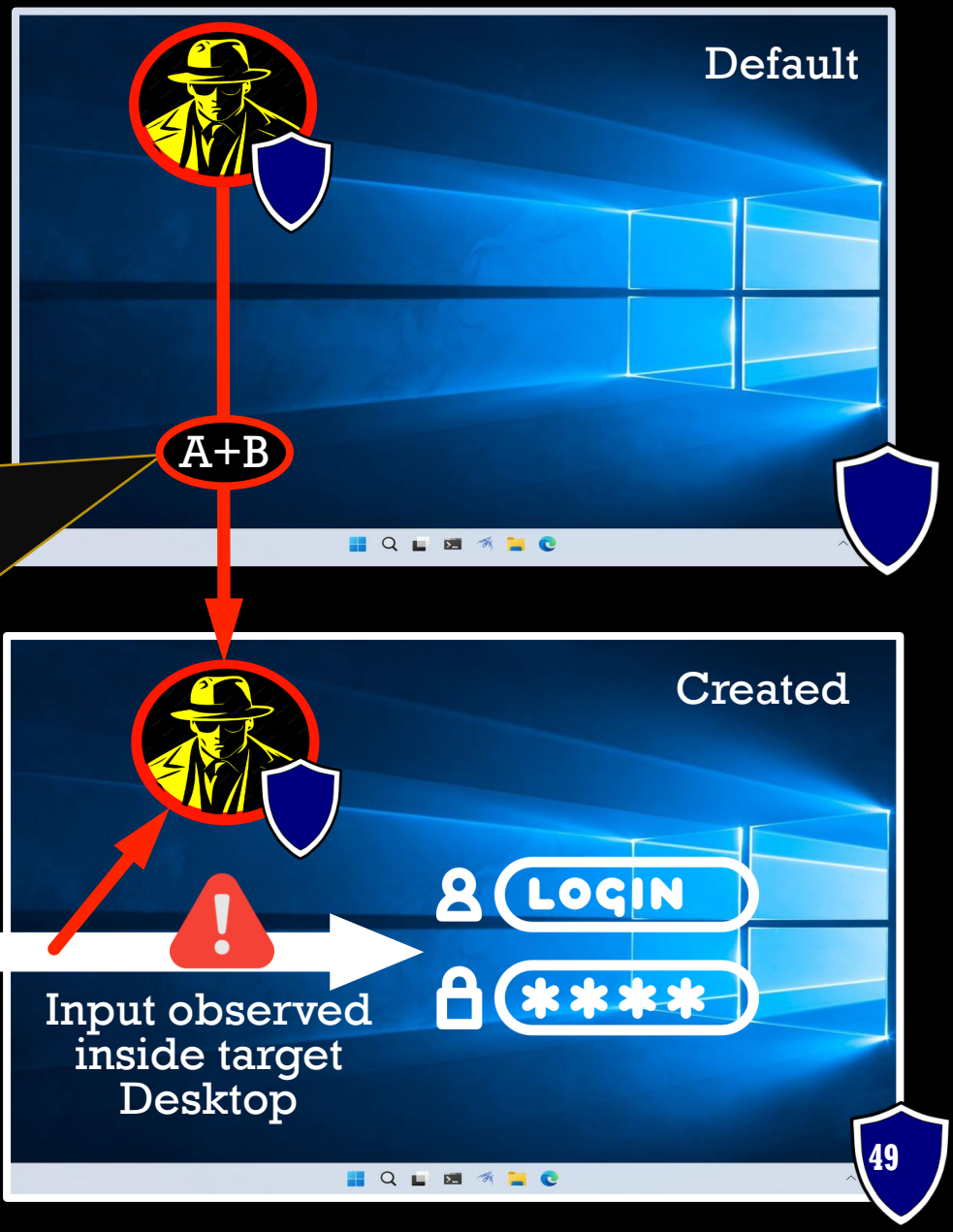


PRIOR ART: THIS ATTACK IDEA ALREADY EXISTS

If same-user attachment succeeds, the keyspy becomes part of the protected context:

Steps	WinAPI
A. Discover	EnumDesktops
B. Open target	OpenDesktop(name)
C. Launch process	CreateProcess + STARTUPINFO.lpDesktop
D. Install hook	SetWindowsHookEx

→ Now attacker can read all the keystrokes

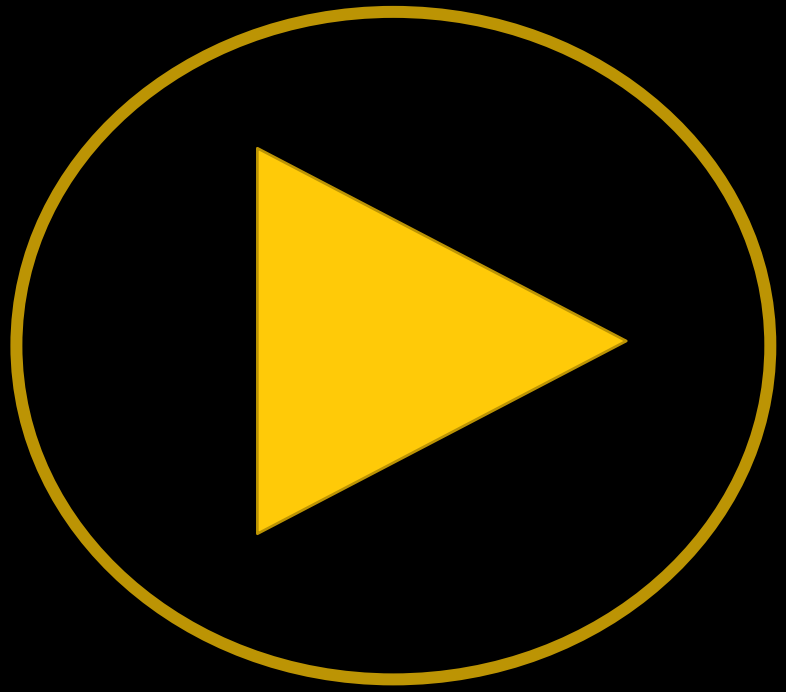


Keylogger vs KeePass Manager

KeepassKeylogger shows that Master Password can be easily keylogged from "Secure desktop"

https://github.com/adenkiewicz/KeepassKeylogger

DEMO #1 – alphaDelta, KeePass, VeraCrypt vs SetWindowsHookEx (Privilege: User)



The online version is here – https://igorkorkin.github.io/

RESULT TABLE

Attacker:

- EnumDesktops + CreateProcess
- SetWindowsHookEx
- **Privilege**: User





Product	Desktop discovered	Launch process	Hook	Result
alphaDelta	☑	☑	☑	**input captured**
KeePass	☑	☑	☑	**input captured**
VeraCrypt	☑	☑	☑	**input captured**

RESEARCH QUESTION #2



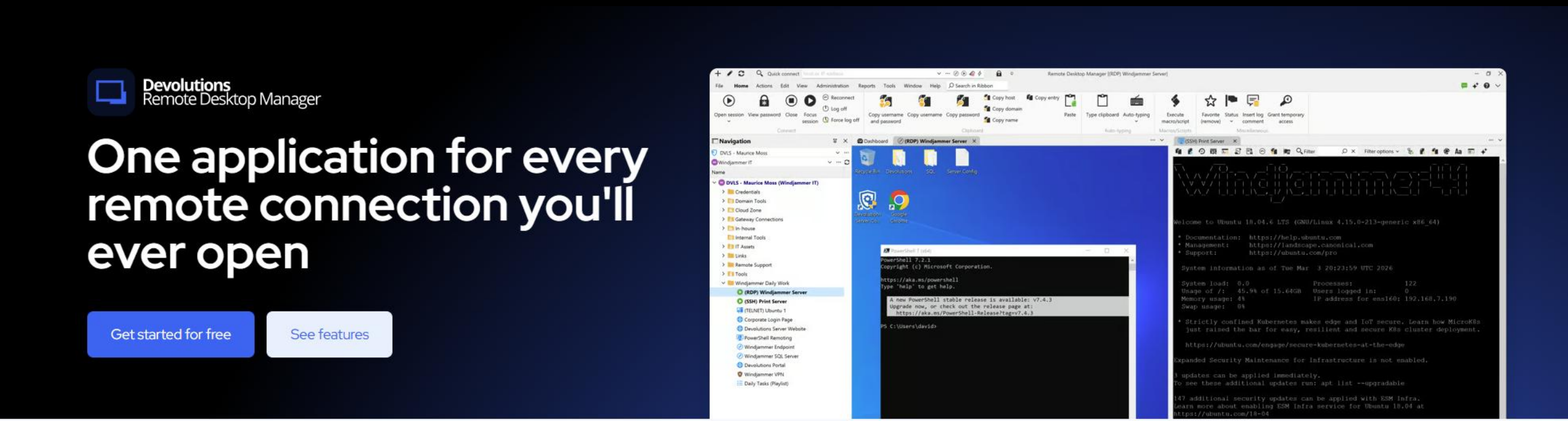
Can the Desktop DACL block a dangerous capability even if the process is allowed to enter?

RESEARCH DISCLAIMER

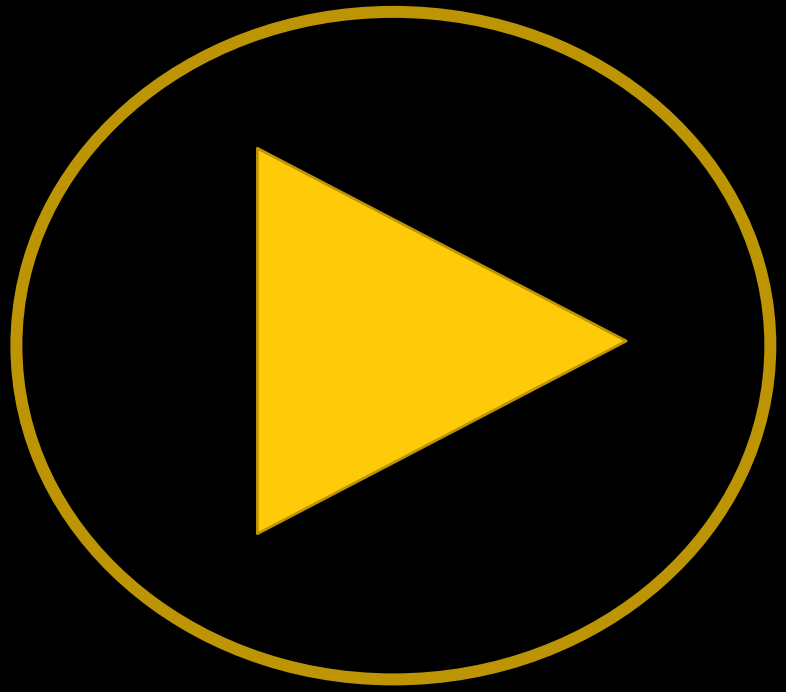
- Lab-only evaluation on our own RDM installation
- Synthetic keyboard input only
- No modification of RDM
- We do not analyze RDM internals.
- We analyze observable Windows Desktop object behavior

Devolutions Remote Desktop Manager

- RDM – a tool for managing remote connections and storing credentials.
- 🔐 Protected UI → separate Windows Desktop



DEMO #2 — RDM vs SetWindowsHookEx (Privilege: User)



The online version is here – https://igorkorkin.github.io/

DEVOLUTIONS RDM: A CASE WITH A RESTRICTIVE SD

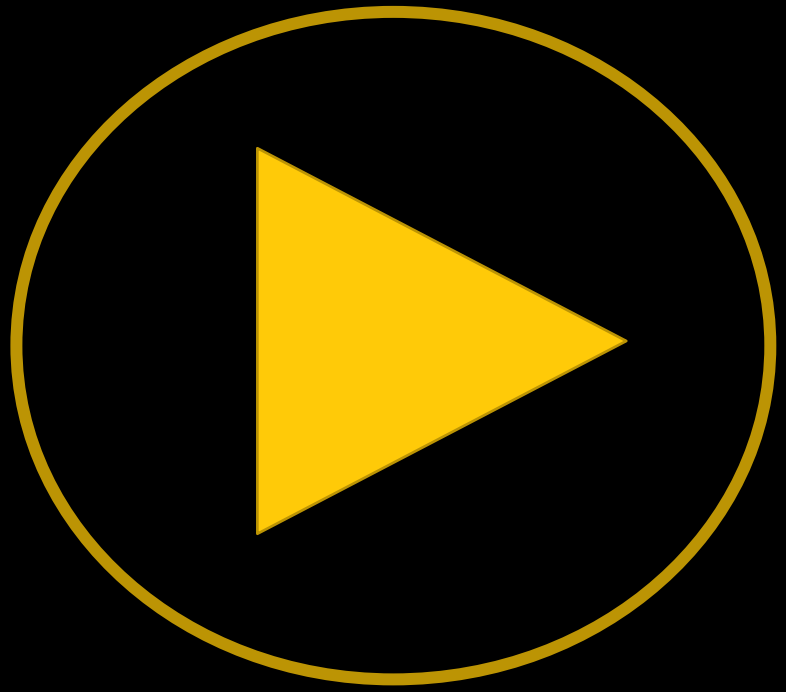
- EnumDesktops: Observed name RDMxxxxx

- Desktop DACL:
  - DENY OWNER RIGHTS 0x00040008
  - ALLOW User 0x00000187

  where 0x00040008 = WRITE_DAC + DESKTOP_HOOKCONTROL

  DESKTOP_HOOKCONTROL is required for hook installation.

DEMO #3 — RDM vs Raw Input (Privilege: User)



The online version is here – https://igorkorkin.github.io/

RESULT TABLE: DEVOLUTIONS RDM



Attacker:

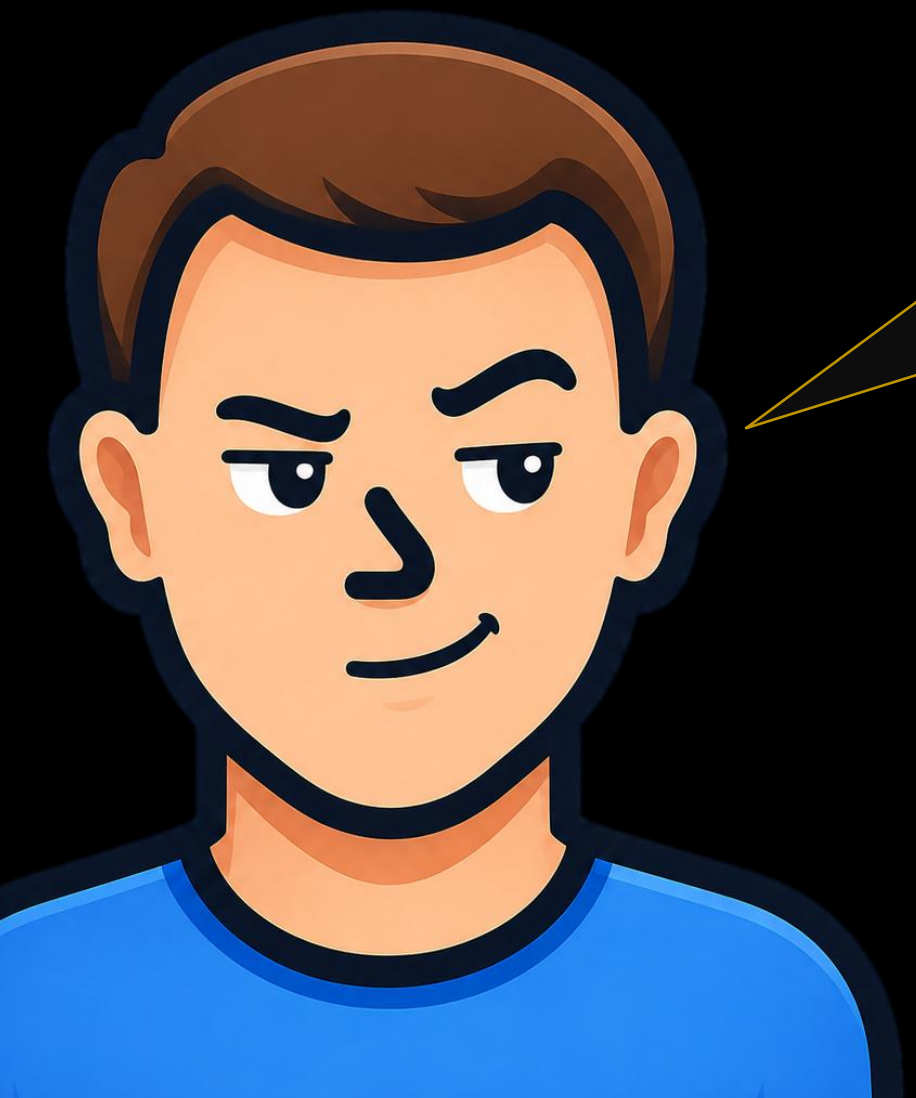
- EnumDesktops + CreateProcess
- SetWindowsHookEx + Raw Input
- **Privilege**: User

Product	Desktop discovered	Launch process	Hook	Raw Input	Result
RDM	☑	☑	✗ ACCESS_DENIED	☑	**input captured**



Desktop access is granular:

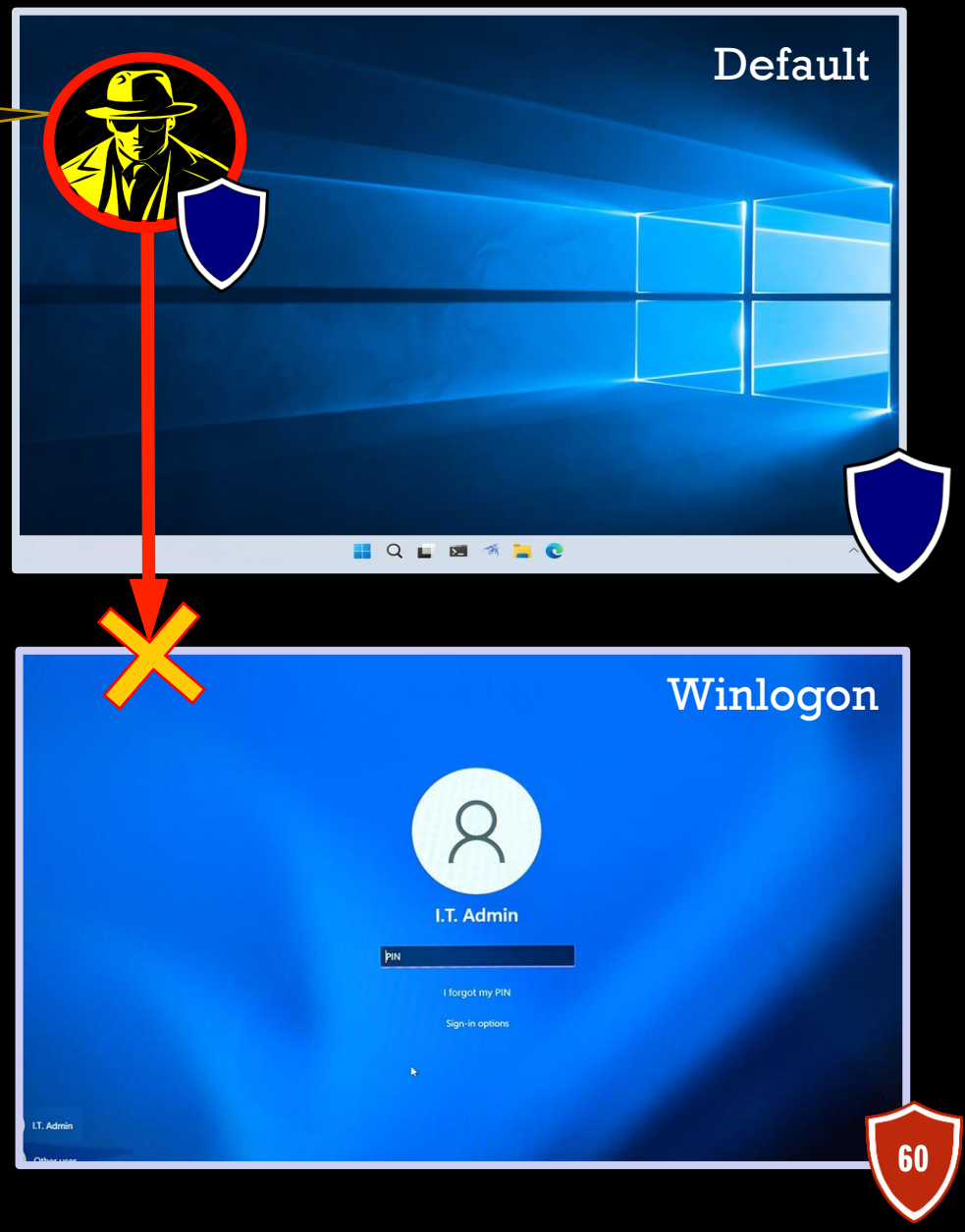
- Blocking one input capability does not necessarily block another input mechanism.
- Raw Input still observed keystrokes inside the RDM Desktop
- No Desktop access-right bit analogous to DESKTOP_HOOKCONTROL for Raw Input.



Winlogon seems protected…
against an ordinary user.

WINLOGON: USER ATTACKER FAILS

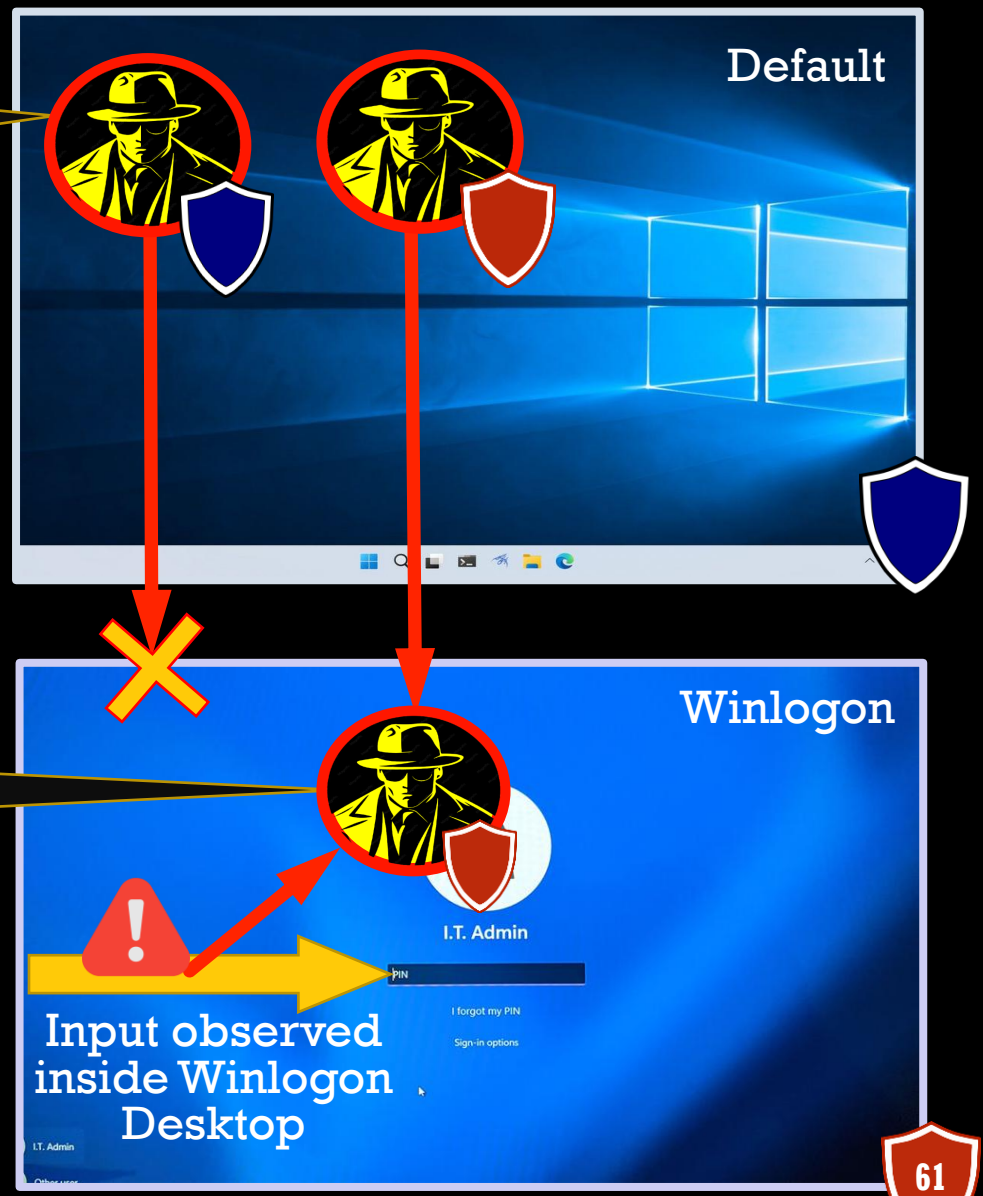
A. User-privileged keyspy cannot attach to Winlogon Desktop.



WINLOGON: SYSTEM ATTACKER SUCCEEDS

A. User-privileged keyspy cannot attach to Winlogon Desktop.

B. SYSTEM-privileged keyspy can attach and install hooks on Winlogon Desktop. (e.g. via PsExec)

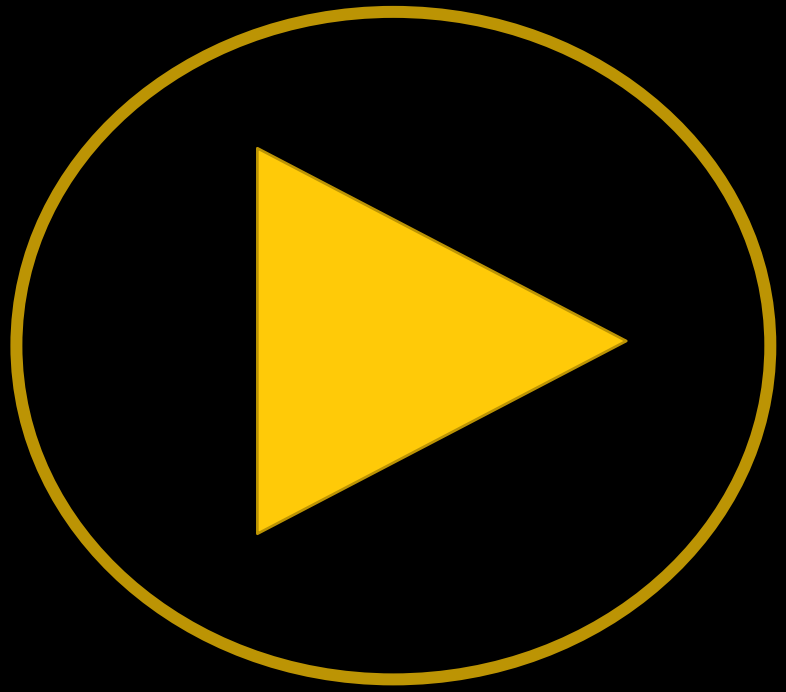


Keyloggers against Winlogon:

1. AdvanceLogger
2. AlmightyKeyLogger

https://github.com/Daniel-Ayz/AdvancedTopicsInMalware/tree/main/AlmightyKeyLogger
https://github.com/yuval2017/AdvanceLogger

DEMO #4 – Winlogon vs SetWindowsHookEx (Privilege: SYSTEM)



The online version is here – https://igorkorkin.github.io/

MATRIX: EXPERIMENTAL RESULTS

	Attacker				**Result**
	Without attaching	With attaching			
Privilege	User	User	Admin	SYSTEM	
Default	SetWindowsHookEx ☑	–	–	–	Input captured

MATRIX: EXPERIMENTAL RESULTS

	Attacker				**Result**
	Without attaching	With attaching			
Privilege	User	User	Admin	SYSTEM	
Default	SetWindowsHookEx ☑	–	–	–	Input captured
VeraCrypt	✗	SetWindowsHookEx ☑	–	–	Input captured
KeePass	✗	SetWindowsHookEx ☑	–	–	Input captured

MATRIX: EXPERIMENTAL RESULTS

	Attacker				**Result**
	Without attaching	With attaching			
Privilege	User	User	Admin	SYSTEM	
Default	SetWindowsHookEx ☑	–	–	–	Input captured
VeraCrypt	✗	SetWindowsHookEx ☑	–	–	Input captured
KeePass	✗	SetWindowsHookEx ☑	–	–	Input captured
Devolutions RDM	✗	• SetWindowsHookEx ✗ • Raw Input ☑	–	–	Input captured

MATRIX: EXPERIMENTAL RESULTS

	Attacker				**Result**
	Without attaching	With attaching			
Privilege	User	User	Admin	SYSTEM	
Default	SetWindowsHookEx ☑	–	–	–	Input captured
VeraCrypt	✗	SetWindowsHookEx ☑	–	–	Input captured
KeePass	✗	SetWindowsHookEx ☑	–	–	Input captured
Devolutions RDM	✗	• SetWindowsHookEx ✗ • Raw Input ☑	–	–	Input captured
Winlogon	✗	✗	✗	SetWindowsHookEx ☑	Input captured

MATRIX: EXPERIMENTAL RESULTS

	Attacker				Result
	Without attaching	With attaching			
Privilege	User	User	Admin	SYSTEM	
Default	SetWindowsHookEx ☑	–	–	–	**Input captured**
VeraCrypt	✗	SetWindowsHookEx ☑	–	–	**Input captured**
KeePass	✗	SetWindowsHookEx ☑	–	–	**Input captured**
Devolutions RDM	✗	• SetWindowsHookEx ✗ • Raw Input ☑	–	–	**Input captured**
Winlogon	✗	✗	✗	SetWindowsHookEx ☑	**Input captured**

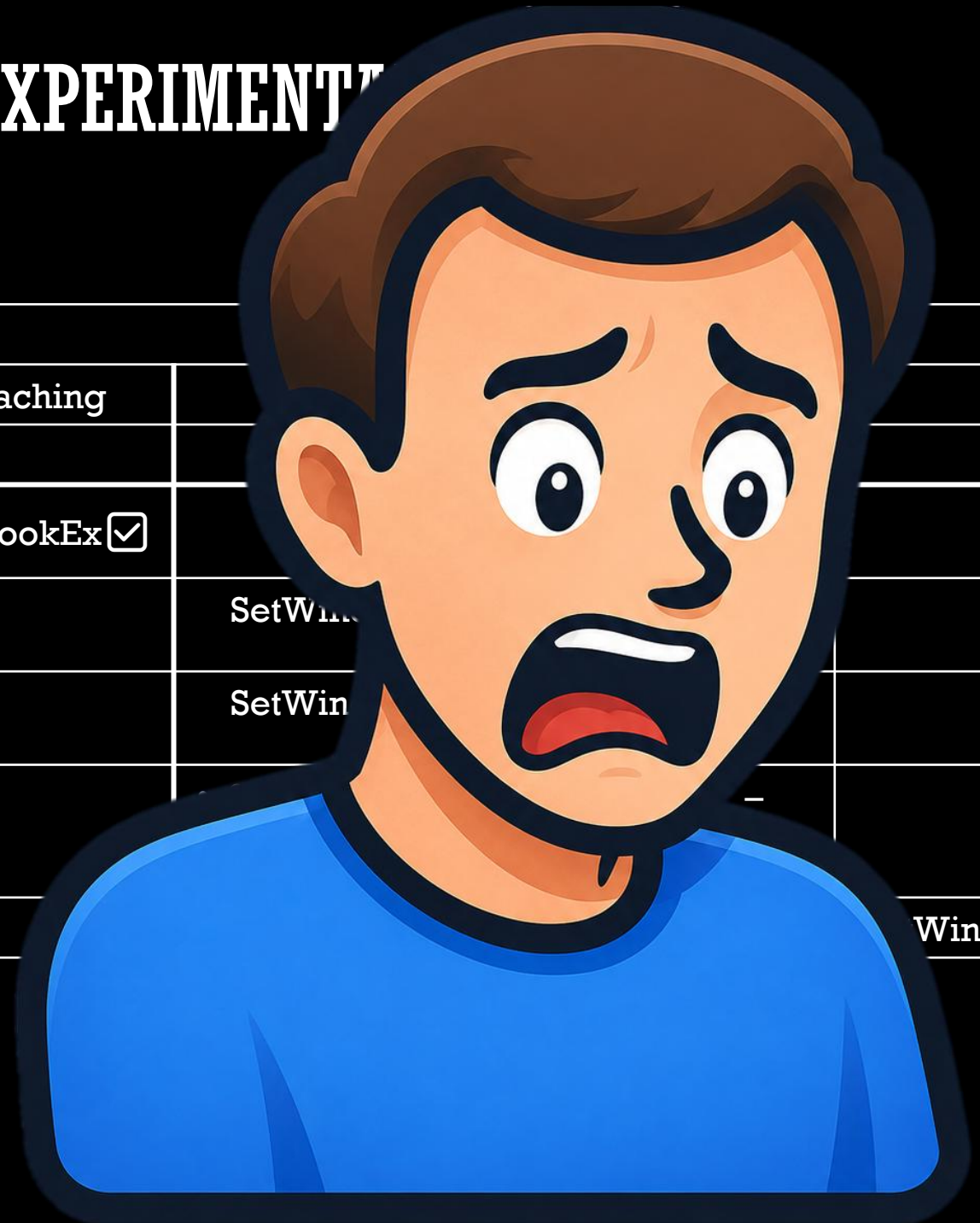
MATRIX: EXPERIMENTAL RESULTS

	Attacker				Result
	Without attaching	With attaching			
Privilege	User	User	Admin	SYSTEM	
Default	SetWindowsHookEx ☑	–	–	–	**Input captured**
VeraCrypt	✗	SetWindowsHo ☑	–	–	**Input captured**
KeePass	✗	SetWindo	–	–	**Input captured**
Devolutions RDM	✗	• SetWindowsHookEx ✗ • Raw Input ☑	–	–	**Input captured**
Winlogon	✗	✗	✗	SetWindowsHookEx ☑	**Input captured**



MATRIX: EXPERIMENTA

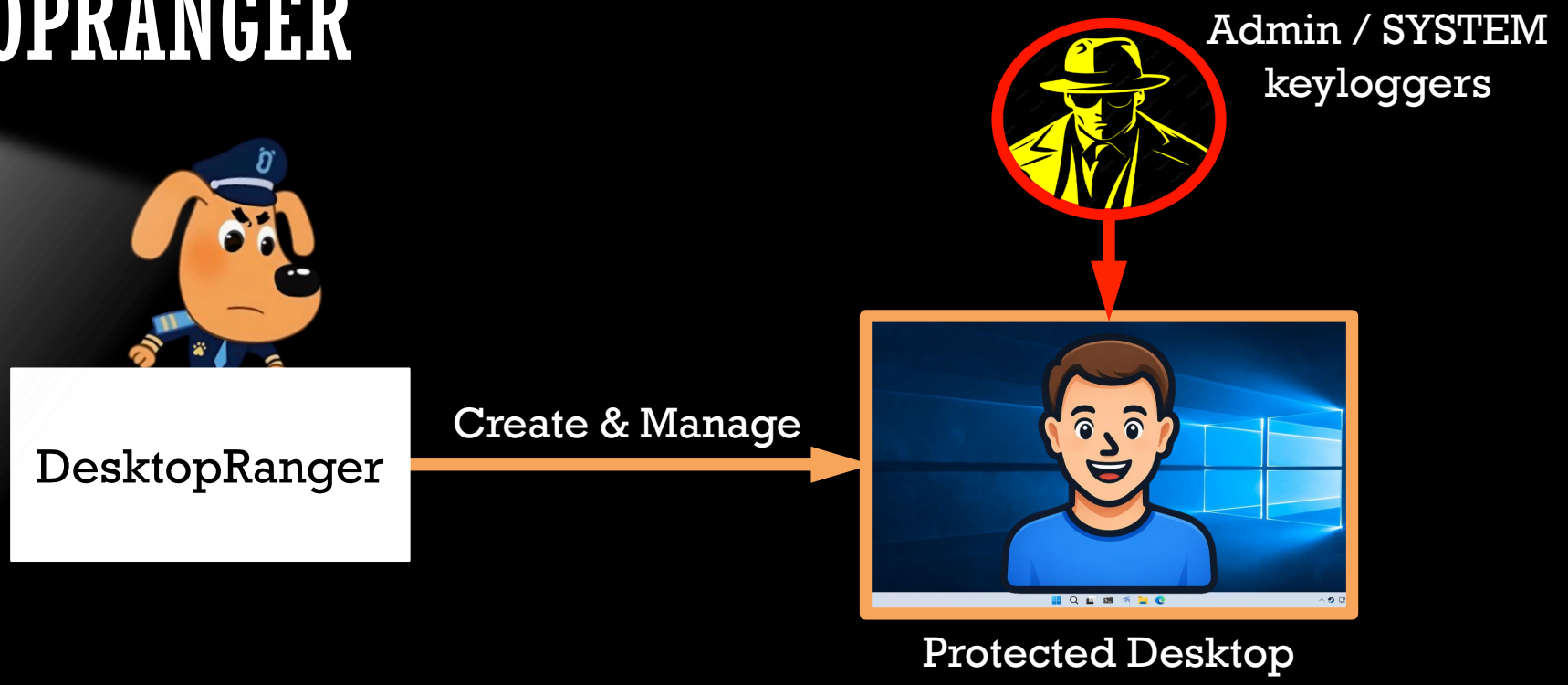
				Result
	Without attaching			
Privilege	User		SYSTEM	
Default	SetWindowsHookEx ☑		–	**Input captured**
VeraCrypt	✗	SetWi	–	**Input captured**
KeePass	✗	SetWin	–	**Input captured**
Devolutions RDM	✗	–	–	**Input captured**
Winlogon	✗		WindowsHookEx ☑	**Input captured**





Can we create a Desktop protected even from high-privilege keyloggers?

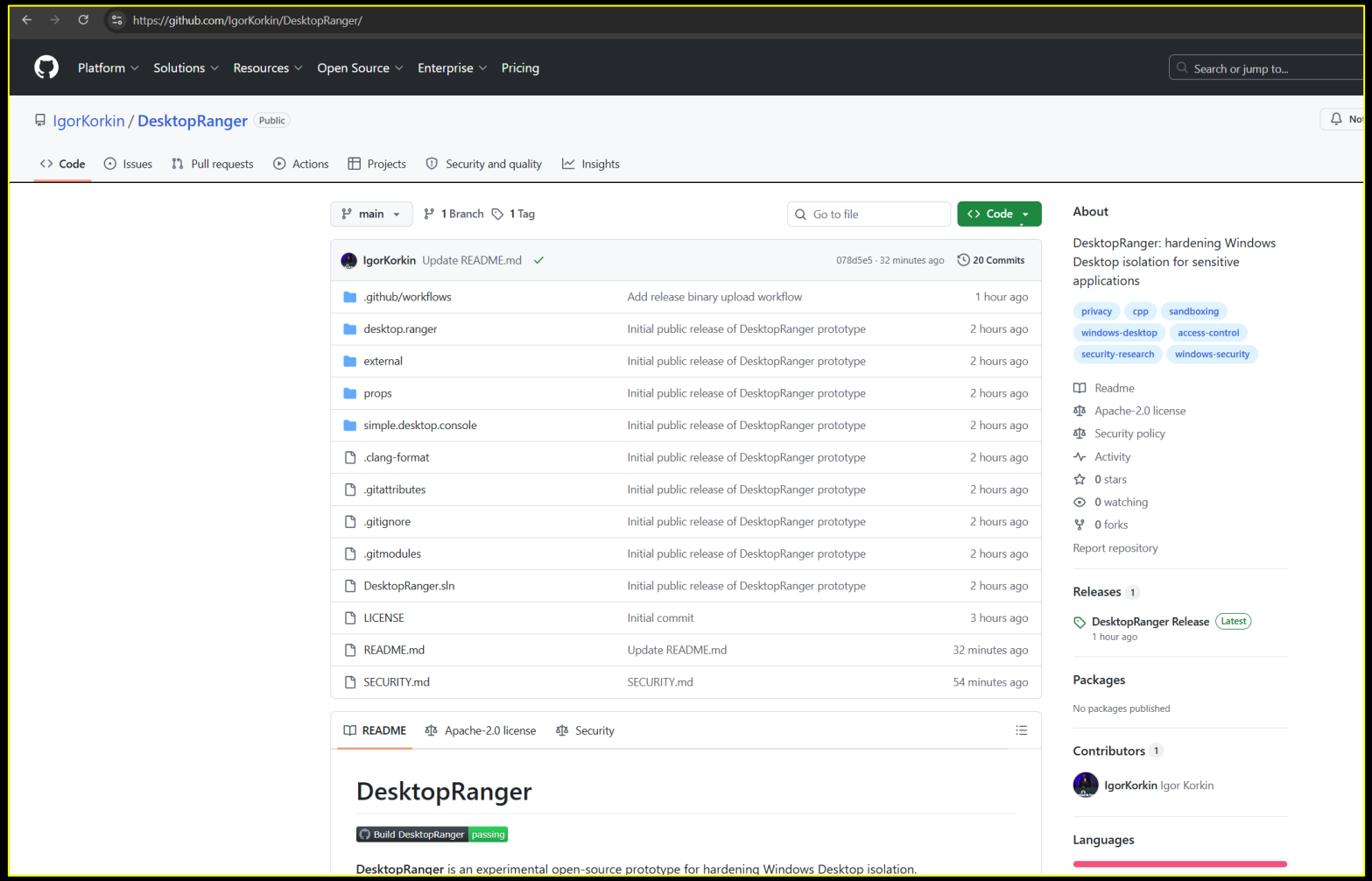
DESKTOPRANGER



- User-mode utility to create a private Desktop for sensitive work
- Blocks enumeration, opening, and unauthorized attachment.
- Targets high-privilege user-mode keyloggers.
- WinAPI + WIL / C++26 / ~3000 SLOC / Windows 11

DesktopRanger: Open-source repository

- A limited public release focused on the defensive prototype: https://github.com/IgorKorkin/DesktopRanger



Step 1 – Create a desktop with an empty DACL



```
CreateDesktopW(RandomName, "D:P")
```

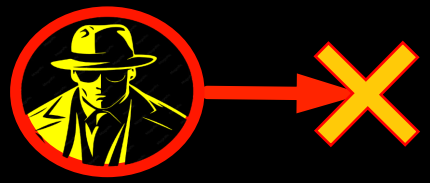
`"D:P"` – protected empty DACL: no ACEs, no inherited access

Step 1 – Create a desktop with an empty DACL



`CreateDesktopW(RandomName, "D:P")`

`"D:P"` – protected empty DACL: no ACEs, no inherited access



WinAPI	Result
EnumDesktops	`RandomName` is not returned
OpenDesktop	Access denied
CreateProcess(`RandomName`)	Access denied

Attacker cannot open the Desktop even as Admin/SYSTEM through WinAPI.



WinAPI	Result
SwitchDesktop/CloseDesktop	☑
CreateProcess	see Step 2



Great! Attacker cannot open the created Desktop!

BUT! How do we launch user apps inside the 2nd desktop?

Step 2 – Temporarily allow access in a secure way to run the app

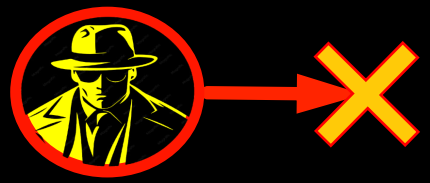


1. Hide Desktop name from enumeration
2. Temporarily allow access
3. Launch trusted app
4. Restore empty DACL: "D:P"
5. Restore enumeration policy

Step 2 – Temporarily allow access in a secure way to run the app



1. Hide Desktop name from enumeration
2. Temporarily allow access
3. Launch trusted app
4. Restore empty DACL: "D:P"
5. Restore enumeration policy



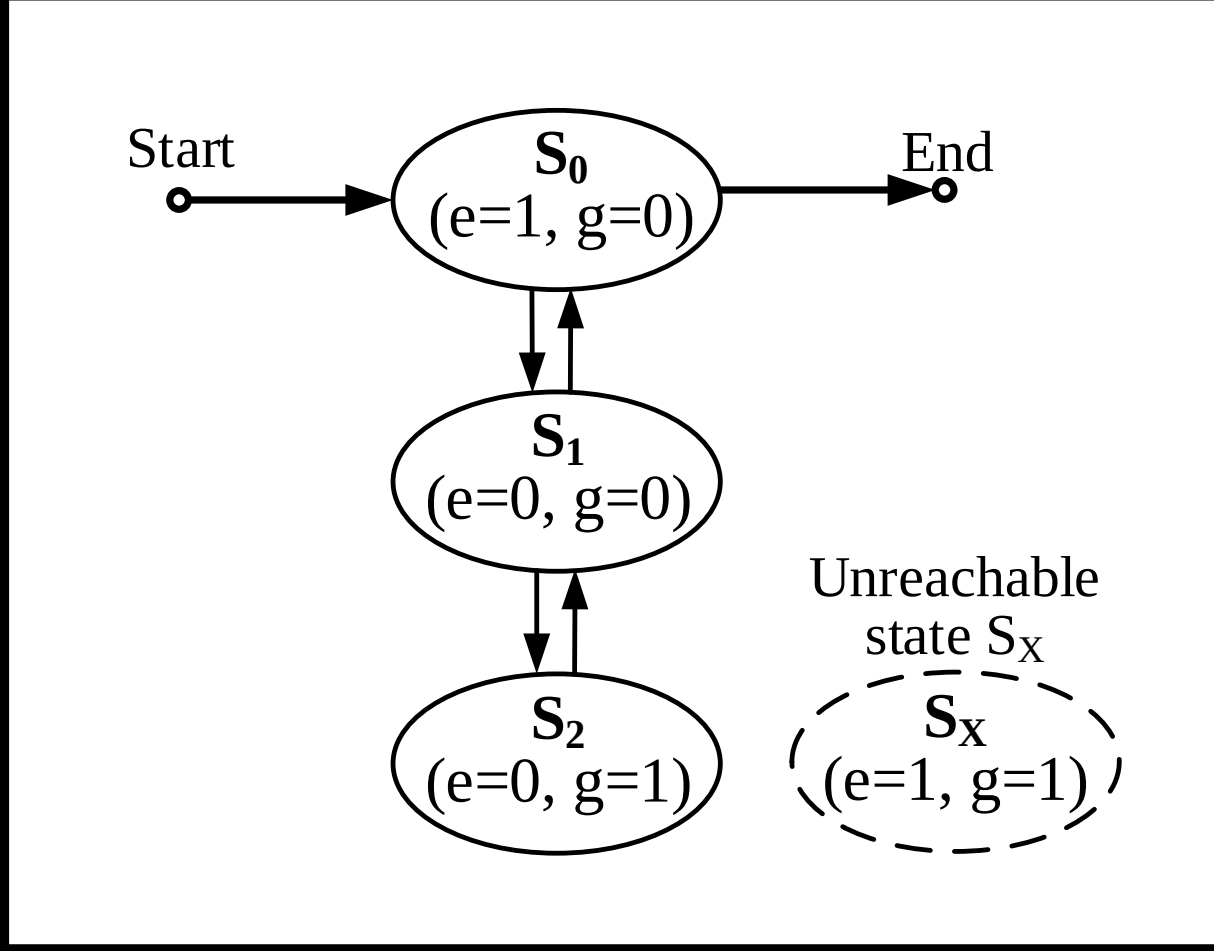
WinAPI	Result
EnumDesktops	Empty list (Step 1)
OpenDesktop(name)	Requires prior knowledge of the random name
n.b. The risky "open" window is short and hidden from enumeration.	



WinAPI	Result
CreateProcess	☑

LIFECYCLE MODEL

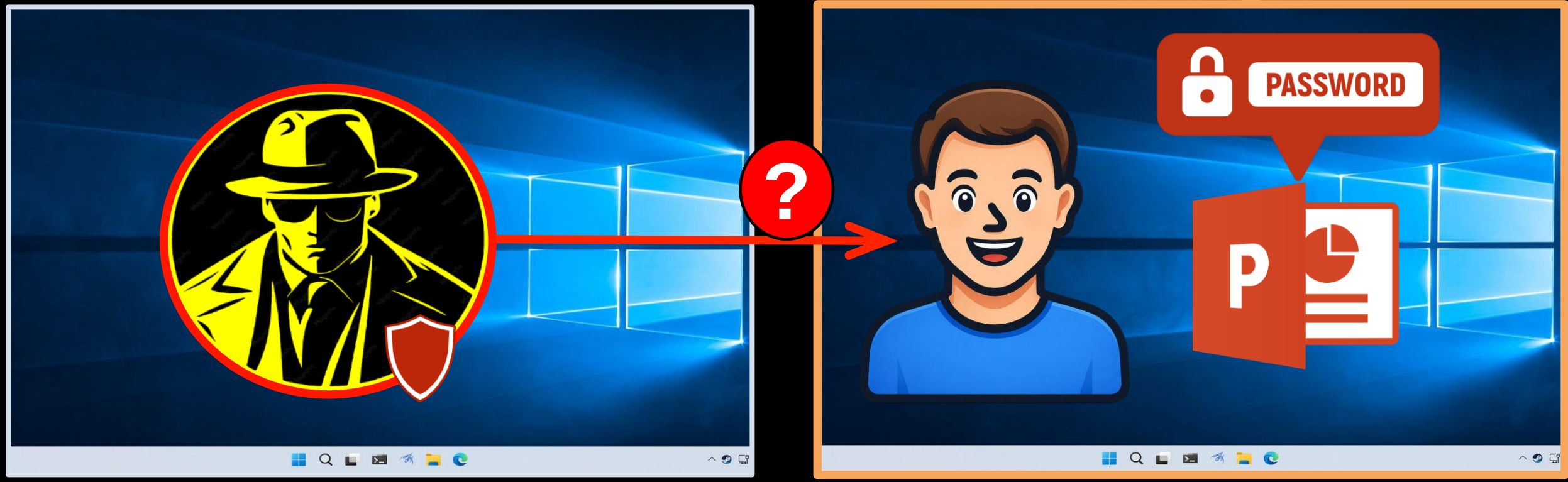
- Variables
  - e — WINSTA_ENUMDESKTOPS enabled
  - g — Desktop open-by-name is granted
- States
  - S0: e=1, g=0 — normal
  - S1: e=0, g=0 — hidden + closed
  - S2: e=0, g=1 — hidden + temporarily open
  - Sx: e=1, g=1 — unreachable state



- Transition graph S0 → S1 → S2 → S1 → S0
- Security invariant: Never allow e=1 and g=1 at the same time.

TESTING SCENARIO

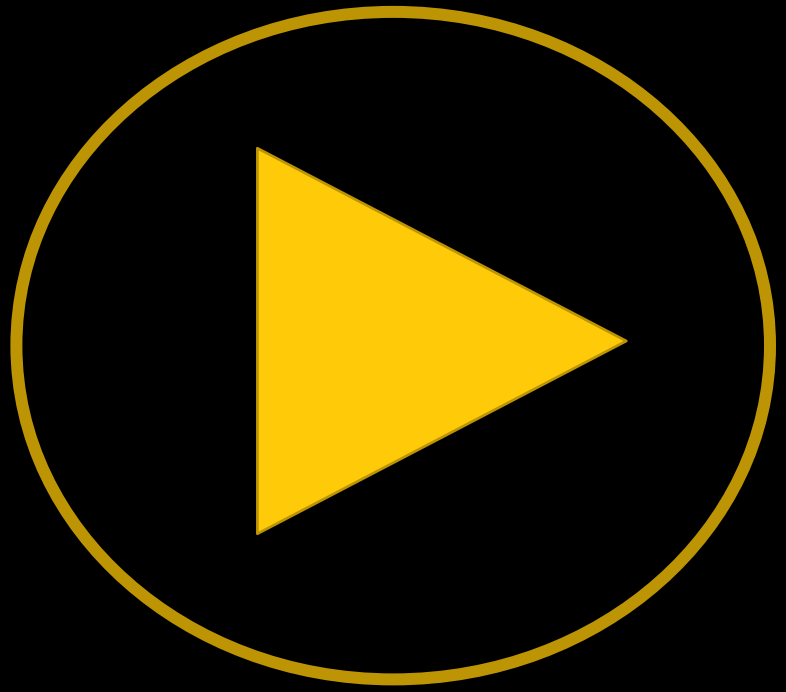




Default Desktop

DesktopRanger

Demo #5 – DesktopRanger vs SetWindowsHookEx (Privilege: SYSTEM)



The online version is here – https://igorkorkin.github.io/

VALIDATION MATRIX

Target	Privilege	Desktop enum/open	Process launch	Hook	Raw	Result
• KeePass • alphaDelta Secure-Desktop • VeraCrypt	User	☑	☑	☑	☑	**Input captured**
Winlogon	SYSTEM	☑	☑	☑	☑	**Input captured**
Devolutions RDM	User	☑	☑	✗	☑	**Input captured**

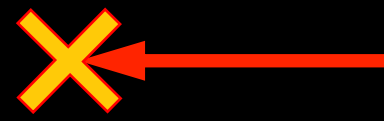
VALIDATION MATRIX

Target	Privilege	Desktop enum/open	Process launch	Hook	Raw	Result
• KeePass • alphaDelta Secure-Desktop • VeraCrypt	User	☑	☑	☑	☑	**Input captured**
Winlogon	SYSTEM	☑	☑	☑	☑	**Input captured**
Devolutions RDM	User	☑	☑	✗	☑	**Input captured**
Desktop Ranger 	System	✗	n/a	n/a	n/a	**Input is protected**

VALIDATION MATRIX

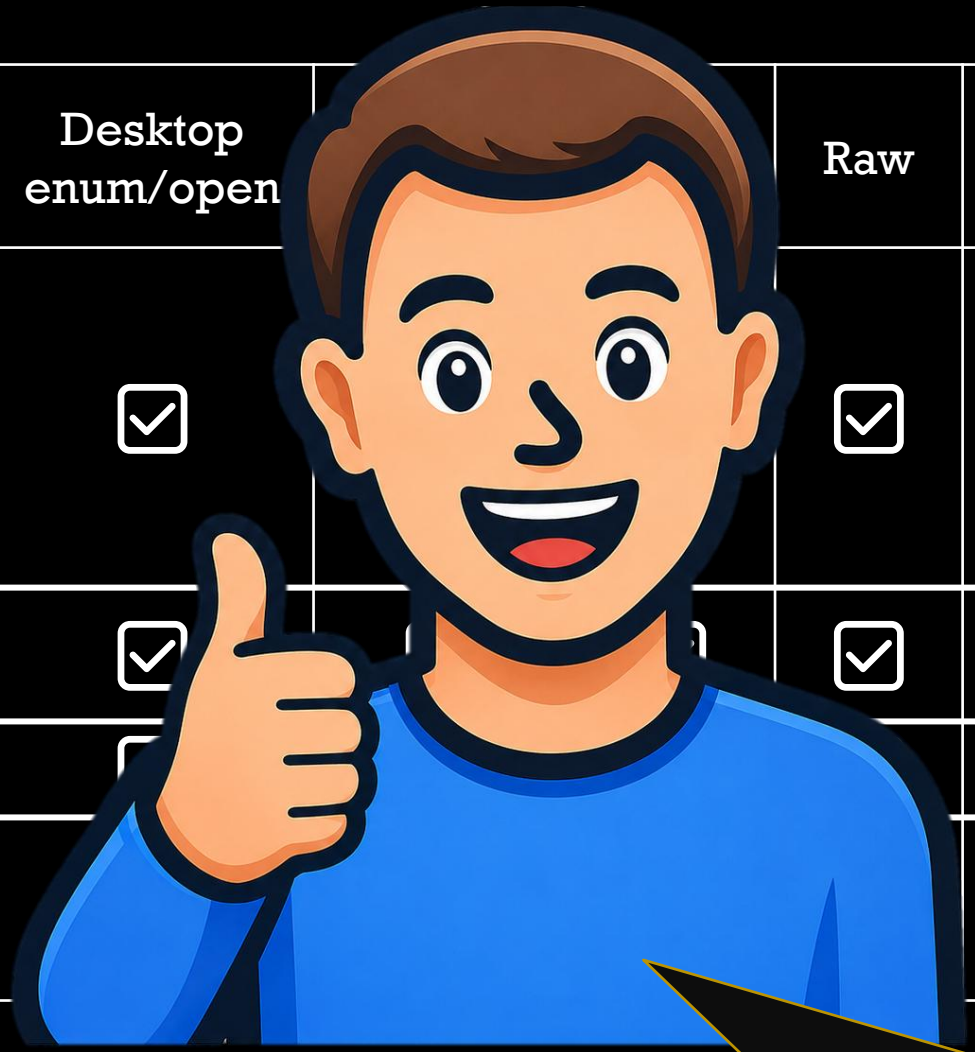
Target	Privilege	Desktop enum/open	Process launch	Hook	Raw	Result
• KeePass • alphaDelta Secure-Desktop • VeraCrypt	User	☑	☑	☑	☑	**Input captured**
Winlogon	SYSTEM	☑	☑	☑	☑	**Input captured**
Devolutions RDM	User	☑	☑	✗	☑	**Input captured**
Desktop Ranger 	System	✗	n/a	n/a	n/a	**Input is protected**

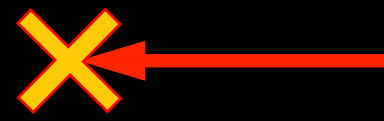




VALIDATION MATRIX

Target	Privilege	Desktop enum/open		Raw	Result
• KeePass • alphaDelta Secure-Desktop • VeraCrypt	User	☑		☑	**Input captured**
Winlogon	SYSTEM	☑		☑	**Input captured**
Devolutions RDM	User				**Input captured**
Desktop Ranger 	System				**Input is protected**





DesktopRanger protects user input from tested user-mode keyloggers.

FIVE TAKEAWAYS

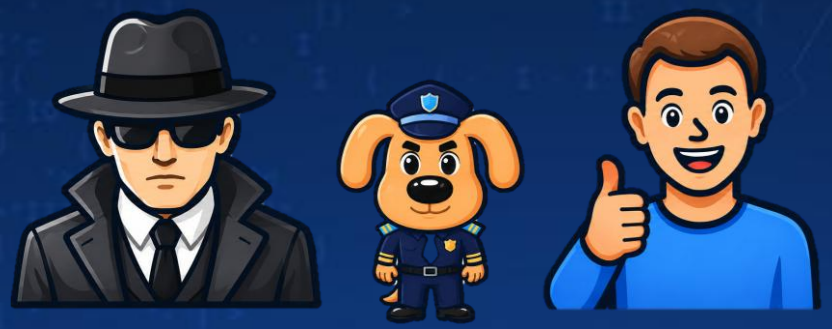
1. Windows Desktop can provide a separate interactive context.
2. Protection depends on Desktop SD/DACL and privileges.
3. Same-user attachment works for several observed implementations.
4. RDM shows granular capability control: Hook ✗, Raw Input ☑.
5. DesktopRanger adds lifecycle-level control through Window Station + Desktop policy.

OFFZONE 2026

THANK YOU! QUESTIONS?

- More research results: igorkorkin.github.io
- DesktopRanger: github.com/IgorKorkin/DesktopRanger







DESKTOPRANGER – DEFENSE IN DEPTH

DesktopRanger 	User-mode App github.com/IgorKorkin/DesktopRanger	Prevent unauthorized attachment: ▪ Window Station + Desktop DACL ▪ Lifecycle control ▪ Documented WinAPI
PrivacyRanger 	Kernel-mode Driver	Enforce attachment policy in kernel: ▪ Control Desktop handle creation ▪ Control attachment attempts ▪ Validate lifecycle transitions
MemoryRanger 	Hypervisor (VT-x) github.com/IgorKorkin/MemoryRanger	Protect Desktop / Window Station state integrity: ▪ Prevent unauthorized attachment ▪ Enforce attachment policy in kernel ▪ Protect security-critical kernel memory

User-mode policy → Kernel enforcement → Hypervisor integrity

Apparently, this research is too cyber for AI

- OpenAI — banned
- Claude — banned

OpenAI - Access Deactivated [C-1RBlwoWE8Nt

noreply@tm.openai.com
to korkiniy

OpenAI

Hello,

OpenAI's terms and policies restrict the use of our services in a number of areas. We have identified activity in ChatGPT that is not permitted under our policies for:

- Cyber Abuse

We are deactivating your access to our services immediately for the account associated with the email korkiniy@gmail.com (User ID: user-FrkOMzKCNr6xfHSpiFTSI748).

If you have questions or think there has been an error, you can use the button below to initiate an appeal.

Initiate appeal

Best,
The OpenAI team

If you have any questions

Chat paused

This request triggered restrictions on violative cyber content and was blocked under Anthropic's Usage Policy. To request an adjustment pursuant to our Cyber Verification Program based on how you use Claude, fill out this form. To learn more about the program or provide feedback, visit our help center. Please start a new chat or retry with Haiku 4.5. If you think Claude's memory of past conversations may have contributed to this, you can clear it in Settings > Memory.

Claude is AI and

Claude

Sign out

Your account will remain closed

We reviewed your account and the additional context you shared, and decided not to restore access. This is our final decision.

Unable to reach the model provider

OpenAI flagged this request for potential high-risk cybersecurity activity. Please try a less sensitive prompt. Learn more here.

Copy Request (c1ffc91b-30b4-42e3-b683-914f3a85ffe7)

Война — это путь обмана.
Вершина искусства войны — побеждать без боя.
Быть невидимым, быть неслышимым —
Хитростью привести противника к поражению.

– Сунь-цзы, «Искусство войны»

Монография знакомит читателя с современными угрозами операционных систем. В центре внимания — перспективные атаки на ядро: техники сокрытия, обход средств защиты и скрытые гипервизоры, использующие аппаратную виртуализацию.

Автор исследует актуальную проблему защиты операционных систем в условиях активного противодействия со стороны нарушителей и предлагает практические пути её решения.

В работе обобщены результаты многолетних исследований автора, прошедшие апробацию на международных конференциях и подтверждённые экспериментальными данными.

Для специалистов по защите информации, преподавателей, студентов и аспирантов, обучающихся по направлению «Информационная безопасность».

Ключевые темы:

- Анализ угроз безопасности ядра Windows: руткиты, вредоносные драйверы, атаки класса BYOVD и гипервизоры-руткиты.
- Практические примеры новых атак на ядро Windows: техники обхода Windows Defender и EDR/AV-решений.
- Описание авторских гипервизорных решений для мониторинга и защиты памяти ядра от атак со стороны драйверов.
- Детали среды для запуска драйверов в изолированных областях памяти (анклавах).
- Математические методы и алгоритмы обнаружения скрытых гипервизоров в условиях противодействия.

Информация об авторе

Игорь Юрьевич Коркин — кандидат технических наук (05.13.19), системный программист и технический эксперт в области информационной безопасности операционных систем и гипервизоров. Автор более 50 научных работ, включая статьи, патент и доклады на международных конференциях в США, Европе, Азии и России: BlackHat, LABScon, HITB, REcon, EKOPARTY, ROOTCON, Texas Cyber Summit, CDFSL, IEEE S&P и РусКрипто. Научный руководитель студентов МИФИ.



ЗАЩИТА ЯДРА ОПЕРАЦИОННЫХ СИСТЕМ В УСЛОВИЯХ ПРОТИВОДЕЙСТВИЯ

Игорь Коркин

Игорь Коркин

ЗАЩИТА ЯДРА ОПЕРАЦИОННЫХ СИСТЕМ В УСЛОВИЯХ ПРОТИВОДЕЙСТВИЯ



Монография:

- 3 раздела, 16 глав
- 100 рисунков, 500 страниц
- Доступна бесплатно в PDF

RESIDUAL NAME GUESSING

- Residual risk: random name guessing during S2-state
- Alphabet = 61
- Length = 255
- $|N| = 61^{255}$

- Practical probability is negligible for the short launch window.

Four user-mode input interception techniques

Technique (keystroke provider)	WinAPI / mechanism	Idea
Hooks	SetWindowsHookEx(WH_KEYBOARD_LL)	callback for keyboard events
Raw Input	• RegisterRawInputDevices() • WM_INPUT • GetRawInputData()	receives raw keyboard events
Polling	GetAsyncKeyState()	periodically checks key state
DirectInput	• DirectInput8Create() • CreateDevice(GUID_SysKeyboard) • GetDeviceData()	reads buffered keyboard-device data

Windows Security architecture already accounts for highly privileged attackers (kernel-level threats)

- **PatchGuard**

  Protects critical Windows kernel code and structures from unauthorized modification. Even highly privileged attackers still face kernel integrity boundaries.

- **Driver Signing · Windows Driver Policy · Vulnerable Driver Blocklist**

  These mechanisms specifically restrict the next escalation step: turning privileged user-mode execution into arbitrary kernel execution, including BYOVD scenarios.

- **VBS / Secure Kernel**

  Moves security-critical components outside the normal Windows kernel trust boundary. Compromising the ordinary OS does not automatically compromise the isolated environment.

- **LKRG — Linux**

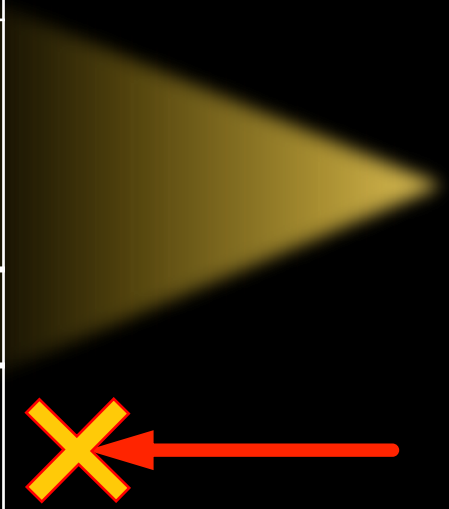
  Detects kernel integrity violations and exploitation attempts at runtime. Linux follows the same principle: high privilege does not eliminate security boundaries.

DesktopRanger applies the same principle to Windows GUI and user input.

VALIDATION MATRIX

Privilege: **User**

Target	Desktop enum/open	Process launch	Hook	Raw	Result
• KeePass • alphaDelta Secure-Desktop • VeraCrypt	☑	☑	☑	☑	**Input captured**
Devolutions RDM	☑	☑	✗	☑	**Input captured**
Desktop Ranger 	✗	n/a	n/a	n/a	**Input is protected**



VALIDATION MATRIX

Privilege: **User**

Target	Desktop enum/open	Process launch	Hook	Raw	Result
• KeePass • alphaDelta Secure-Desktop • VeraCrypt	☑	☑	☑	☑	**Input captured**
Devolutions RDM	☑	☑	✗	☑	**Input captured**
Desktop Ranger 	✗	n/a	n/a	n/a	**Input is protected**

Privilege: **SYSTEM**

Winlogon	☑	☑	☑	☑	**Input captured**
Desktop Ranger 	✗	n/a	n/a	n/a	**Input is protected**

DesktopRanger: Sandbox-Driven Endpoint Security

- **Complements AV/EDR**

  AV/EDR detects malicious activity. DesktopRanger limits what an attacker can do even before detection.

- **Isolation by Default**

  Modern security is moving from one large trusted environment toward multiple small isolated environments.

- **GUI-Level Isolation**

  DesktopRanger brings this principle to the Windows GUI and protects interactive user input..

- **Zero Trust Inside**

  High local privileges should not automatically grant access to every security-sensitive resource inside the OS.

- **Near-Zero Overhead**

  No continuous interception or event processing is required. Protection is enforced primarily through native Windows isolation and access-control mechanisms.

Private sandboxes should become the default endpoint model.