隧道之外
VPN 保護邊界的量測

VPN BOUNDARY – MEASUREMENT AND RESULTS

隧道之外

VPN 保護邊界的量測方法與實測結果

Odysec 研究報告 WP-002 ｜ **版本** 1.0 ｜ **發布日** 2026-08-31 **領域**：網路 × 實體（Cyber × Physical）

目錄

關於本報告的性質

本報告的所有數字皆來自一個完全公開的受控實驗環境。該環境的拓撲檔與全部腳本 隨本報告一同發布，任何人都能重建並重跑。

受測對象是手動組態的 WireGuard（`wg-quick`），不是任何商業 VPN 應用程式。 商業產品有自己的 kill switch、DNS 處理與重連邏輯，本報告對它們一無所知。凡在 本報告中看到的數字，其適用範圍僅止於第 3 章所描述的那個環境。這條界線在第 8 章 再次說明，撰寫引用時請勿越過它。

本報告量測的是**隧道與路由邏輯的失效模式**，屬於方法論驗證，不是產品評測。

摘要

「VPN 會洩漏」不是新聞。`AllowedIPs` 不含 `::/0` 就不承載 IPv6、不指定解析器 就沿用本地那台——這些都是文件裡就有的既定行為。真正缺的不是「會不會」，而是 **保護邊界的形狀**：在哪些時刻、哪些流量、洩漏多少、持續多久。

本報告以一個四節點的受控環境，對八種情境、共十七個組態各執行 30 次，合計 510 次量測， 得到四個主要發現：

1. **決定會不會洩漏的不是「連線是否中斷」，而是「隧道介面是否還在」：** 實體連線中斷兩秒再恢復，全部 30 次執行**零洩漏**——因為隧道介面與其路由在 斷線期間依然存在，封包被送進隧道後丟棄，不會改道實體介面。反之，當隧道介面 本身消失（對應 VPN 行程異常結束、被作業系統回收、休眠喚醒後未重建），**每一個封包都以真實位址送出，而且不會自行停止**。危險的不是訊號不穩， 是 VPN 程式悄悄停掉——而那時網路完全通暢，畫面上沒有任何徵兆。
2. **失效分成四類，對策完全不同：** 狀態類（建立空窗、抖動、介面消失）靠 kill switch 解決，而且**不付出延遲代價**：有無 kill switch 的隧道建立時間 統計上沒有差異，它只是把「暴露」換成「封包被丟棄」。組態遺漏類（IPv6、DNS） 則 kill switch 完全幫不上忙，只能靠正確設定。第三類最不易察覺—— captive portal 例外放行本地網段，**kill switch 全程正常運作，流量卻仍然出得去**。 第四類則根本不是「洩漏」，見下一點。
3. **洩漏與否是組態的確定結果，不是機率事件：** 在 IPv6、DNS 與 portal 例外 三個情境中，洩漏比例的離散度**為零**——30 次執行的結果完全相同。IPv6 在 僅承載 v4 的隧道下是 100% 洩漏、DNS 在未接管解析器時是 100% 洩漏， 沒有任何一次例外。這不是運氣問題，是組態決定的必然結果。 （時間類量測則有離散度，見第 4 章的四分位距。）
4. **把一切都設定對，本地網路仍然知道你是誰：** 在全隧道、解析器接管、 嚴格 kill switch 全程有效的組態下，不可信的本地網路**每一次都仍然取得** 這台裝置的網卡位址、網卡廠商、主機名稱與本地位址（30 次執行完全相同）。 這些流量不是「漏出隧道」，而是**從來就沒有進過隧道**——ARP 與 DHCP 甚至不走一般的 IP 送出路徑，因此 kill switch 在原理上碰不到它們。 隧道保護的是它被設定去承載的東西，不是「你在這個網路上的存在」。

行動建議：若您依賴 VPN 隱藏所在位置，請確認三件事——`AllowedIPs` 是否涵蓋 IPv6（或已停用 IPv6）、解析器是否確實由隧道接管、以及 kill switch 是否在登入 公共網路後移除了本地網段例外。這三項各自獨立，做對兩項不會補償漏掉的第三項。**而若您還需要這個網路認不出這台裝置，那需要的是裝置層的對策，不是任何 VPN 設定。**

---

1. 問題陳述

1.1 這件事對誰重要

對多數人而言，VPN 洩漏是隱私問題。對本團隊前兩篇報告所關注的讀者——正在 躲避跟蹤者的人——它是人身安全問題。

WP-001《貼身的眼睛》的方法論從「第零原則：改用一部乾淨裝置」開始。但乾淨 裝置連上網路之後呢？若該裝置的真實位址或 DNS 查詢逸出隧道，落入加害者可觸及的 網路，前面所有的謹慎都會在那一刻歸零。**一次洩漏就足以暴露所在位置。**

這正是「數位風險轉為實體風險」的典型路徑，也是本報告接續前兩篇的原因。

1.2 缺的不是「會不會」，是邊界的形狀

VPN 的洩漏行為在技術社群中討論已久，但公開的論述多半停在「有可能洩漏，請小心」 的層次；中文圈的內容則以產品評測為主，鮮少有系統性的失效測試。

這種論述對讀者的幫助有限。「有可能」無法回答下列任何一個問題：

- 按下連線之後、保護真正生效之前，有多長的空窗？期間送出了什麼？
- 走進電梯訊號中斷再恢復，這段期間會不會洩漏？
- VPN 程式若被作業系統回收，我會知道嗎？流量會停還是會裸奔？
- kill switch 要付出什麼代價？它擋得住哪些情況、擋不住哪些？

這些都是可量測的。本報告的工作就是把它們量出來。

1.3 本報告要回答的問題

1. 在各種失效情境下，客户端的真實位址、DNS 查詢與流量是否逸出隧道？
2. 若洩漏，範圍多大、持續多久？
3. kill switch 在哪些情境有效、哪些無效、代價為何？
4. 這些失效能否被使用者察覺？

---

2. 威脅模型

2.1 攻擊者

本報告假設的攻擊者**控制被害者所在的本地網路**：家用路由器、飯店或機場的 Wi-Fi、或加害者提供的行動熱點。這是跟蹤情境中極常見的位置——共同居住者、 前伴侶、或掌握住所網路設備的人，都自然具備這個位置。

攻擊者不需要入侵任何裝置，也不需要破解加密。他只需要**看見**：看見哪些封包 以真實位址送出、看見哪些網域被查詢。

2.2 能力假設

攻擊者能夠：

- 觀察本地網段上的全部流量與其來源位址
- 提供 DHCP 設定，包括指定本地 DNS 解析器
- 觀察並記錄所有經由該解析器的查詢
- 在本地網段上架設服務

攻擊者**不能**：

- 解密 WireGuard 隧道內的內容
- 於受測裝置上執行程式碼
- 取得隧道端點的控制權

也就是說，本報告談的完全是**旁路**問題：加密沒有被破解，只是有些東西根本 沒有進入加密通道。

2.3 被害者處境

延續 WP-001 的假設：被害者已察覺自身可能遭受監控，正採取防護措施， 且**技術知識有限**——他遵循一般建議安裝了 VPN，看到介面顯示「已連線」， 便認為位置已受保護。

本報告要檢驗的正是這個認知與實際狀態之間的落差。

3. 實驗方法

3.1 環境與拓撲

實驗於 EVE-NG 網路模擬環境中進行，四個節點皆為 Debian 13，透過序列主控台 腳本化操作，全程無人工介入。

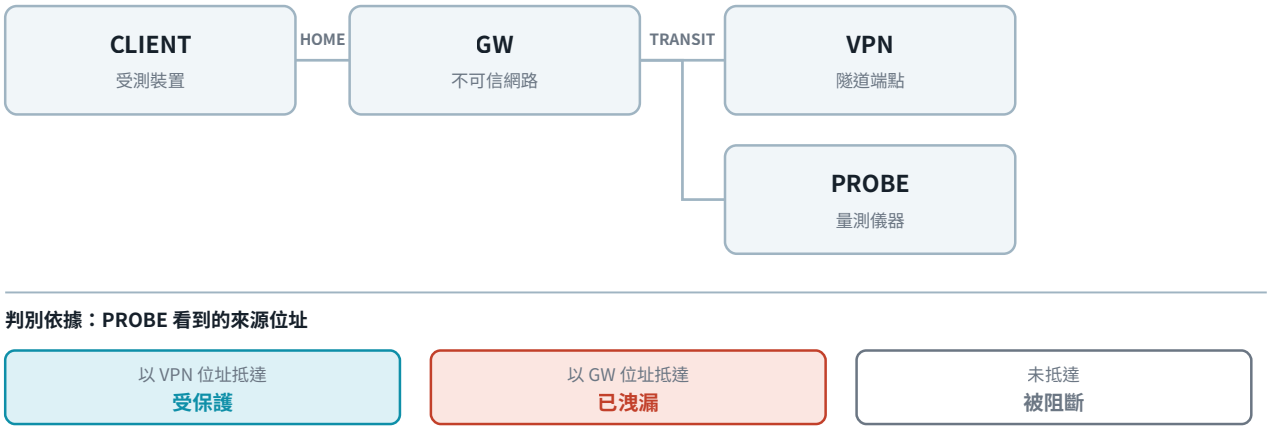

圖 1：實驗拓撲與判別依據

節點	角色
CLIENT	受測裝置。以 DHCP 取得不可信網路的設定，建立 WireGuard 隧道
GW	家用路由器／不可信的本地網路。DHCP、本地 DNS 解析器、對外 NAT
VPN	隧道端點。隧道流量由此以其自身位址送出
PROBE	量測儀器。權威 DNS 與 HTTP／UDP 接收端，記錄每個請求的來源位址

位址全部採用 RFC 5737 與 RFC 3849 的文件用範圍，讀者重建時不會撞到真實網段。

3.2 判別原理

整套量測只依賴一件事：**PROBE 觀察到的來源位址**。

觀察到的來源	判定
VPN 的位址	流量經隧道送出，受保護
GW 的位址	已洩漏——繞過隧道直接送出
未抵達	被阻斷（kill switch 生效）或遺失

這個判別成立的前提是**兩個網段之間沒有旁路**。實驗前已驗證：在 GW 尚未設定 NAT 時，CLIENT 無法以任何方式抵達 TRANSIT 段。若沒有這項驗證，來源位址就無法 唯一對應到路徑，整套量測失去意義。

3.3 信標設計

以 20 毫秒間隔的 UDP 信標量測時間窗。選擇 UDP 而非 HTTP 是刻意的：UDP 無連線 狀態、無重傳、無握手，送出與否忠實反映當下的路由與過濾狀態。**TCP 的重傳會把 時間軸抹平，量不到窗口邊界。**

信標與擾動（例如「發出連線指令」）跑在**同一個行程的同一個時鐘**上。若分屬 兩個行程，就得處理行程啟動抖動——實測數十毫秒，與待測的窗口同一個數量級。 所有時間運算都在客户端自己的單調時鐘上完成，PROBE 只負責歸屬來源，因此完全 不需要跨節點對時。

每個信標序號的結局只有三種：以 VPN 位址抵達、以 GW 位址抵達、未抵達。 這三種涵蓋了所有待測情形。

3.4 情境與對照

八個情境，每格重複 30 次，共 510 次執行。s1 至 s6 各有兩個變體； s7 另加一個補充變體以便歸因（見 4.4）。

情境	擾動	基準組	對照組
s1 建立空窗	發出連線指令	無 kill switch	有 kill switch
s2 實體抖動	連線中斷 2 秒後恢復	無 kill switch	有 kill switch
s3 介面消失	刪除隧道介面	無 kill switch	有 kill switch
s4 IPv6	穩態（無擾動）	`AllowedIPs` 僅 v4	納入 `::/0`
s5 DNS	穩態（無擾動）	不接管解析器	隧道接管解析器
s6 portal 例外	隧道未建立	帶本地網段例外	嚴格 kill switch
s7 範圍之外	穩態（無擾動）	未使用 VPN	全隧道＋接管＋嚴格 kill switch （補充組：同組態但無 kill switch）
s8 stub 解析器	穩態（無擾動）	未指定解析器	指定解析器

每個變體只改一件事。若一個變體同時動兩處，量出差異也無從歸因。s4 的 伺服器端一律雙堆疊備妥，受測變因只有客户端的 `AllowedIPs`。

s3 的擾動選擇值得說明。第一版設計以「移除隧道對端」模擬失效，但那樣量不到 東西：介面還在、路由仍指向它，封包只是被靜靜丟棄，**不會洩漏**。真正危險的是 介面連同路由一起消失，流量隨即回退到原本的預設路由。**選錯擾動方式會量到 假的陰性。**

3.5 儀器驗證

實驗過程中遭遇三次**靜默失效**——儀器已經壞掉，但表現得與真實的陰性結果 一模一樣。若未察覺，測出的「沒有洩漏」會被當成結論。

1. **客户端拿不到 IPv6：**節點映像為了路由器角色預設開啟 IPv6 轉發， 而 Linux 在轉發開啟時會拒收路由器公告。客户端因此完全沒有 IPv6 位址—— IPv6 洩漏測試會量到「沒有洩漏」，但那是因為根本沒有 v6 可洩漏。
2. **路由器送不出路由器公告：** `ip addr flush` 會連同 `fe80::` link-local 位址一起清除且不會自動恢復，而公告必須以 link-local 為來源。**失敗完全 靜默**：DNS 服務照樣把每次公告記入日誌，看日誌以為一切正常，實際上封包 擷取顯示線路上只有客户端的請求、沒有任何回應。
3. **設定套用了但沒有作用：**隧道組態中的解析器設定經由系統的解析器管理 機制生效，而本實驗為了讓「客户端問了誰」可直接觀察，已停用該機制。 結果是設定看起來套用了、解析器卻沒變，對照組與基準組完全一樣。

由此得到一條通則，也建議任何進行類似量測的人採用：

> **每個情境開跑前，先驗證儀器本身處於可量測狀態，而不是直接看結果。**

三項驗證已寫成負向檢查納入佈建腳本——例如 link-local 沒有恢復就直接中止， 不讓實驗在壞掉的儀器上跑下去。

---

4. 實測結果

以下數字均為 30 次執行的中位數，括號內為四分位距。採用中位數而非平均值， 是因為這類延遲的分布右偏，平均值會被少數離群值拉走。

4.1 狀態類失效

情境	無 kill switch	有 kill switch
s1 發出連線指令→保護生效	53.3 ms（52.8–71.9），其後仍洩漏 **2 個封包**（最差 4）	53.4 ms（53.2–53.5），洩漏 **0**
s2 連線中斷 2 秒後恢復	恢復後 47.8 ms 重新受保護，洩漏 **0**	43.2 ms，洩漏 **0**
s3 隧道介面消失	**每一個封包都洩漏**，且不會自行停止	事件前 8.6 ms 起完全靜默，洩漏 **0**

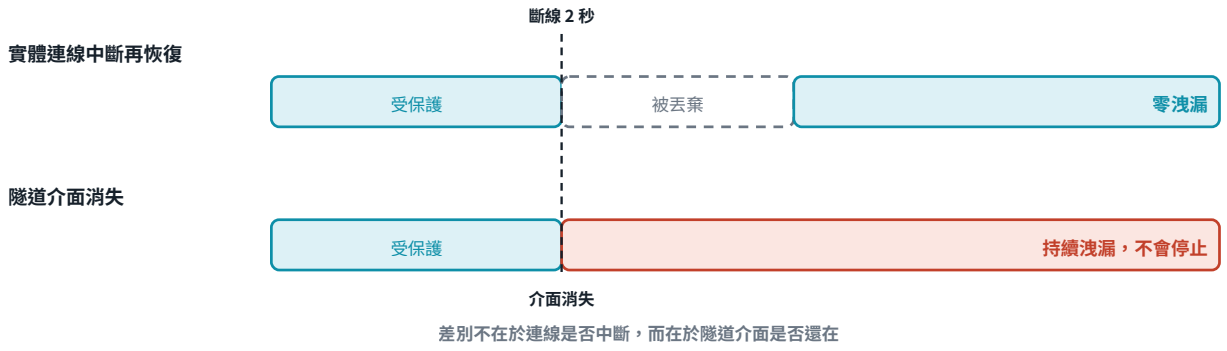

圖 2：兩種失效的時間軸對照

三項各有值得注意之處。

s1：使用者按下連線之後，保護並非立即生效。在此環境中約 53 毫秒的建立 期間，仍有 2 至 4 個封包以真實位址送出。使用者主觀上「已經連上了」，實際上 還沒有。

s2 是本報告最反直覺的結果。實體連線中斷再恢復，30 次執行**全部零洩漏**。 原因是隧道介面與其路由在斷線期間依然存在：封包被送進隧道後因對端不可達而 丟棄（可觀察到約 102 個序號缺口），**不會改道實體介面**。切換網路、進入電梯、 訊號中斷都屬於這一類。

s3 則相反。當介面本身消失，路由回退到實體介面，此後每一個封包都以真實 位址送出，而且**不會自行停止**——實驗窗口結束時仍在洩漏。對應的真實事件是 VPN 行程異常結束、被作業系統回收、或休眠喚醒後未重建。

4.2 組態遺漏

情境	基準組（常見預設）	對照組（正確設定）
s4 IPv6 流量	僅 v4 隧道 → **100% 洩漏**（30 次全部如此）	納入 `::/0` → 0.7%
s5 DNS 查詢	不接管解析器 → **100% 洩漏**（30 次全部如此）	接管 → **0%**

兩個基準組的離散度都是零。**這不是機率問題，是組態決定的必然結果。**

s4 對照組殘留的 0.7%，是隧道建立前送出的那 2 個封包——與 s1 量到的尾巴 是同一件事，不是 IPv6 特有的問題。

s8：stub 解析器之下的同一個問題。 s5 為了讓「客户端問了誰」可直接觀察， 刻意停用了系統的 stub 解析器；但真實桌面預設是啟用的。s8 在啟用的條件下重測， 每格 30 次、每次 10 個不重複的名稱（每格 300 個查詢）：

量測項	未指定解析器	指定解析器
只走隧道的名稱	0	**10 / 10**
只走本地解析器的名稱	**10 / 10**	0
同時送往兩邊	0	0

沒有部分洩漏，也沒有雙送。 在此組合下，隧道的解析器接管完全有效：其機制 是在隧道介面上設定涵蓋全部網域的路由網域，stub 解析器因而不再向其他介面分送 查詢。這一點值得說明，因為 stub 解析器的 split DNS 機制原本正是可能造成 「部分查詢仍走本地」的來源——本實驗未觀察到該情形。

4.3 例外本身

captive portal 例外是為了讓使用者能登入飯店或機場的入口網頁；登入時隧道 尚未建立，因此幾乎所有 kill switch 實作都會放行本地網段。

變體	抵達不可信網路的封包	完全阻斷
嚴格 kill switch	**0**	30/30 次
帶 portal 例外	**226**（30 次皆同）	0/30 次

觀測點設在 GW 上——攻擊者控制的正是本地網段，凡是抵達那裡的封包都已經落入 他手中。誘導方式在真實世界是 DNS：本地解析器在例外之下仍可存取，攻擊者只要 回答一個本地網段的位址即可。

kill switch 全程正常運作，流量卻仍然出得去。這是八個情境中最不易察覺的 一個：使用者看到 kill switch 已啟用，而它確實啟用了。

4.4 範圍之外

前三類失效的共同形狀是「該進隧道的流量沒進去」。第四類不同：**有一整類流量 從來就沒有進過隧道**，因為隧道從未被設定去承載它們——其中有些甚至不走 IP 的 一般送出路徑。

觀測點設在 GW，也就是不可信的本地網路那一側。三種組態各 30 次：

觀測項	未使用 VPN	一切正確的組態	補充：同組態但無 kill switch
客户端送出的訊框（中位數）	23	6	22
可識別資訊項目	10	**4**	9

三格的可識別資訊項目數**離散度皆為零**——30 次執行的結果完全相同。

在「一切都設定正確」的組態下**仍然外洩的四項**，以及 kill switch 擋不住的原因：

仍可取得的資訊	為何擋不住
網卡位址（MAC）	乙太訊框一律帶著真實 MAC；嚴格 kill switch 之下唯一送得出去的雖然只剩隧道自己的封包，那些訊框同樣帶著它
網卡廠商（OUI）	由 MAC 前三個位元組直接得出
主機名稱	DHCP 以明文夾帶主機名稱，而該請求走原始通訊端，不經 IP 送出鏈
本地位址	ARP 屬於不同的過濾家族，同樣不經 IP 送出鏈

補充變體讓成因可以逐項歸屬：

- **隧道本身只移除了一項**（NTP）。它是一般的路由 IP 流量，被全隧道設定一併 承載。其餘九項，隧道連碰都沒碰到。
- **另外五項是 kill switch 移除的，不是隧道**：mDNS、LLMNR 與 IPv6 的路由器 請求、鄰居請求、群播監聽報告都走 IP 送出路徑，因而撞上阻擋規則。
- **四項兩者都擋不住**，理由如上表。

這一類的意義：kill switch 全程有效、隧道全程存在、解析器也確實接管—— 不可信的本地網路仍然知道這台裝置是誰。它不是「洩漏」，而是承載範圍的邊界； 也正因如此，前三類的對策對它全部無效。

> **本節不能支持的推論**：本實驗只測試了一種 kill switch 實作（作用於 IP 送出 鏈的規則集）。**不得據此推論「沒有任何 kill switch 擋得住 ARP 與 DHCP」**—— 作用於網路裝置層送出鉤點的規則集未經測試。此外，可識別資訊的清單受限於本 客户端所安裝的軟體：本實驗的 DHCP 客户端預設不送出供應商類別與客户端識別碼， 故此處量到的是「未送出」而非「無此管道」；其他作業系統會送出。

4.5 綜合

流量／事件	預設組態下的結果
IPv4 穩態	受保護
IPv4 實體連線抖動	受保護
IPv4 隧道介面消失	**100% 洩漏，且不會停止**
IPv4 剛按下連線的數十毫秒內	洩漏 2–4 個封包
IPv6 全部	**100% 洩漏**
DNS 查詢全部	**100% 洩漏**
portal 例外之下	**100% 逃逸**（即使 kill switch 正常運作）
本地網段的身分資訊	**一切正確仍外洩四項**（MAC、OUI、主機名稱、本地位址）

---

5. 討論

5.1 四類失效，四種對策

類別	情境	對策	kill switch 有效?
狀態	建立空窗、抖動、介面消失	kill switch	✔ 有效且不付代價
組態遺漏	IPv6、DNS	正確設定	✘ 完全無效
例外本身	portal 放行本地網段	登入後移除例外	✘ 例外使其失效
範圍之外	ARP、DHCP 主機名稱、乙太層的 MAC	換裝置層的對策：隨機化 MAC、不送主機名稱	✘ 碰不到

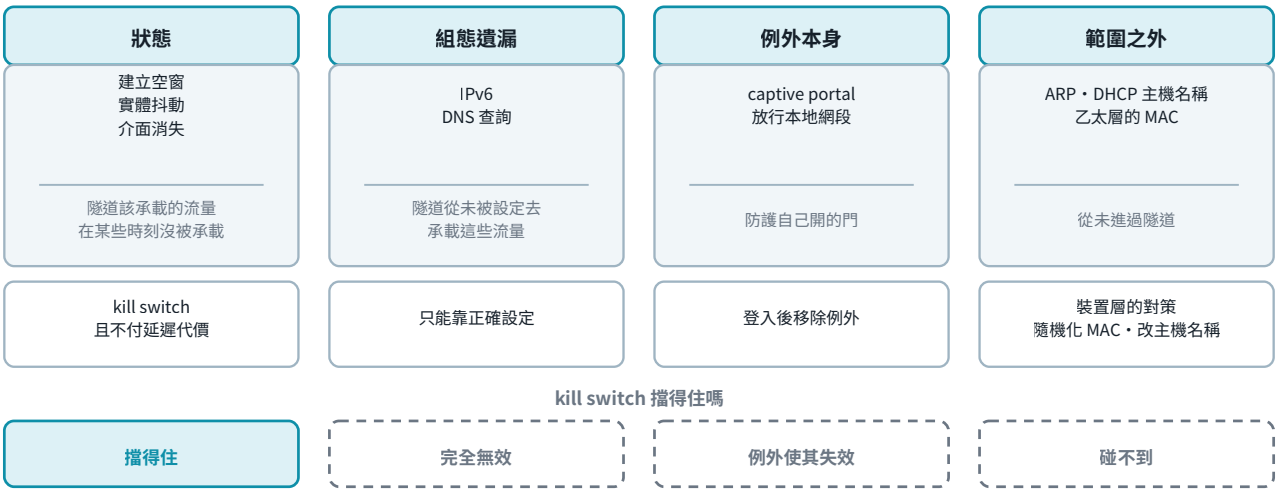

圖 3：四類失效與各自的對策。kill switch 的作用範圍只涵蓋第一類。

這個區分是本報告最主要的實用產出。它把「VPN 安不安全」這個含糊的問題， 拆成四個對策各不相同的具體問題。**做對其中三項，不會補償漏掉的第四項。**

前三類的差別在於「隧道有沒有把該承載的承載好」，第四類的差別在於 **那些資訊從一開始就不在隧道的職責範圍內**。把 VPN 當成「隱身」的人， 最容易誤解的正是這一類：他以為買到的是不被認出，實際上買到的是流量內容與目的地不被看見。

5.2 kill switch 的代價比想像中低

一個常見的顧慮是 kill switch 會拖慢連線。實測顯示：有無 kill switch 的隧道 建立時間**統計上沒有差異**（s1：53.3 對 53.4 毫秒；s2：47.8 對 43.2 毫秒）。

它並不讓連線變慢，而是把「暴露」換成「封包被丟棄」。在 s1 中，原本會外洩的 封包改為在本機被擋下；在 s3 中，它是唯一能阻止洩漏的機制。

5.3 為什麼「斷線才危險」的直覺是錯的

多數人對 VPN 洩漏的想像是「連線斷掉的瞬間會裸奔」。s2 的結果顯示並非如此： 只要隧道介面存在，斷線期間的封包會被丟棄而不是改道。

真正危險的是介面消失，而那對使用者是**不可見的**——網路完全通暢，網頁照常 開啟，沒有任何徵兆。使用者唯一可能察覺的線索，反而是那些他不會注意的地方： VPN 應用程式的圖示狀態、或系統通知欄裡消失的隧道標記。

這個落差本身就是風險：**使用者用「還能不能上網」判斷保護是否有效，而這個 判準在最危險的情境下恰好給出錯誤答案。**

---

6. 給使用者的檢查清單

以下各項各自獨立，請逐項確認。

一、IPv6 是否在隧道之內 確認隧道組態的允許範圍同時涵蓋 IPv4 與 IPv6。若您的 VPN 不支援 IPv6， 較安全的做法是在作業系統層停用 IPv6，而不是任其在隧道之外流動。

二、解析器是否由隧道接管 連線後確認系統實際使用的 DNS 伺服器已改為隧道提供者，而非本地網路派發的那台。**設定檔裡寫了不等於生效**——本研究即遇到設定看似套用、解析器卻未 改變且無任何錯誤訊息的情況。請以系統實際狀態為準。

三、kill switch 的例外是否已移除 若您為了登入公共 Wi-Fi 的入口網頁而放行了本地網段，**登入完成後請移除該例外**。 在例外存在期間，kill switch 對本地網路是無效的。

四、本地網路仍然認得出這台裝置 以上三項全部做對之後，不可信的本地網路仍會取得您的網卡位址、網卡廠商、 主機名稱與本地位址。若您的威脅模型包含「不希望這個網路認出這台裝置」， 需要的是裝置層的對策——隨機化網卡位址、把主機名稱改為無識別性的字串—— **而不是任何 VPN 設定** 。

此外，請理解一項限制：**您無法從「網路是否通暢」判斷保護是否有效。**最危險的 情境下網路完全正常。若情況攸關人身安全，請在每次重要操作前主動確認對外位址， 而不是依賴應用程式的狀態顯示。

---

7. 給實作者與廠商的建議

1. **介面消失應被視為失效事件並主動告警：**目前的行為是靜默回退，使用者 毫無所覺。這是本研究中危害最大且最易修補的一項。
2. **portal 例外應具備自動失效機制：**例如限定時間、或在隧道建立後自動撤除。 讓例外無限期存在，等於讓 kill switch 在最需要的場合失效。
3. **解析器接管應可驗證且失敗時應報錯：**設定套用失敗卻不產生任何訊息， 會讓使用者在錯誤的安全感下操作。
4. **建立期間的封包應預設阻斷：**實測顯示這不會增加建立延遲。
5. **kill switch 的作用範圍應誠實揭露：**作用於 IP 送出鏈的規則集擋不到 ARP 與 DHCP。若產品的介面把 kill switch 呈現為「阻斷一切流量」，那是與實作不符的 描述。要把這兩者也納入，規則必須下到網路裝置層的送出鉤點。
6. **不應把使用者的主機名稱交給不可信的網路：**DHCP 的主機名稱選項以明文送出， 而多數使用者不知道自己的裝置正在廣播一個可識別的名稱。以隱私為訴求的產品 宜提供「連線時隨機化網卡位址與主機名稱」的選項，並說明其限制。

---

8. 研究限制與後續工作

8.1 外部效度界線（最重要的一項）

本報告的受測對象是手動組態的 `wg-quick`，不等於任何商業 VPN 應用程式。

商業產品普遍實作自己的 kill switch、解析器接管與重連邏輯，其行為可能與本 實驗完全不同——可能更好，也可能更差。本報告**沒有**測試任何商業產品，也 **不應**被引用為對任何產品的評價。

本報告能夠支持的推論是：這些失效模式在協定與路由層面**是可能發生的**，且其量測方法是可行且可重現的。要知道某個特定產品的行為，必須對該產品另行 測試——那屬於不同性質的工作，需要真實裝置而非模擬環境。

8.2 量測解析度

信標間隔為 20 毫秒，所有毫秒數值帶有此量化誤差。s1 的 53.3 毫秒實際落在 約 33 至 53 毫秒之間。縮短間隔可提高解析度，但會加大量測本身對系統的擾動。

8.3 環境差異

實驗於模擬網路中進行，鏈路延遲遠低於真實網路。真實環境中的建立延遲會更長， 因此本報告的時間窗數值應視為**下限**而非典型值。

s1 中「整次執行洩漏 53 個封包」的數字未列入結論，因為那是實驗參數（連線前 1 秒基線）的產物；真實世界中該段長度取決於使用者多久才連線，可能是數秒， 也可能是整個開機期間。

8.4 後續工作

- 對真實裝置上的商業 VPN 應用程式進行同類量測
- 作用於網路裝置層送出鉤點的 kill switch 能否涵蓋 ARP 與 DHCP（4.4 節未測）
- 其他作業系統的 DHCP 客户端所送出的可識別欄位（供應商類別、客户端識別碼）
- 行動平台的休眠與喚醒行為
- 量測環境本身的封裝，讓非技術使用者也能自測

8.5 我們希望收到的回饋

若您以本報告的環境測試了其他組態或產品並得到不同結果，我們希望知道。 若您認為本報告的方法有缺陷，我們同樣希望知道——實驗環境完全公開，正是 為了讓這種檢驗成為可能。

附錄 A：自行重現

實驗環境的拓撲檔與全部腳本隨本報告發布，可自下列位址下載：

https://odysec.org/research/wp-002-lab-vlk-274.tar.gz

解開後為 `vlk-274/` 目錄。

環境僅使用可自由取得的 Debian 13 映像，**刻意不使用任何不可合法散布的授權 映像**，因此讀者能夠完整重建。四份佈建腳本皆為冪等，可重複執行；整套環境 在本研究期間曾從零重建三次。

重現步驟、量測工具用法與紀錄格式詳見該目錄的說明文件。

附錄 B：完整數據

每次執行的原始結果以 JSON Lines 格式收於上述封存檔的 `vlk-274/data/`，共 510 筆。彙整腳本 `aggregate.py` 可重現本報告的所有統計量並輸出 CSV。

附錄 C：名詞對照

中文	English	日本語
隧道	tunnel	トンネル
隧道介面	tunnel interface	トンネルインターフェース
洩漏	leak	リーク
中斷開關	kill switch	キルスイッチ
允許範圍	allowed IPs	許可 IP 範囲
解析器	resolver	リゾルバ
解析器接管	resolver takeover	リゾルバの引き継ぎ
路由器公告	router advertisement	ルーター広告
信標	beacon	ビーコン
建立空窗	establishment gap	確立までの空白
入口網頁	captive portal	キャプティブポータル
對照組	control group	対照群
離散度	dispersion	ばらつき
外部效度	external validity	外的妥当性
範圍之外	out of scope	範囲外
可識別資訊	identifying information	識別可能な情報
網卡位址	MAC address	MAC アドレス
網卡廠商代碼	OUI	OUI
主機名稱	hostname	ホスト名
送出鏈	output chain	出力チェーン
原始通訊端	raw socket	生ソケット

附錄 D：資料來源

本報告的數據全部來自本團隊自建的受控實驗環境，無外部資料來源。協定行為的 描述依據 WireGuard 官方文件與 `wg-quick` 手冊；位址範圍依據 RFC 5737 與 RFC 3849。

本報告與 WP-001《貼身的眼睛：近距離監控的偵測與應對》屬同一系列，威脅模型一致。

引用格式：Odysec（2026）。《隧道之外：VPN 保護邊界的量測方法與實測結果》。Odysec 研究報告 WP-002，版本 1.0。

免責聲明：本報告為技術研究與教育用途，不構成法律意見。個案的法律評價請諮詢執業律師。若您正處於人身安全的立即危險，請撥打 110；保護服務與諮詢專線為 113。