基于控件隐藏的主机安全测试技术研究

任益辰[1] 汪美琴[1] 刘明哲[1] 董国超[1]
REN Yichen WANG Meiqin LIU Mingzhe DONG Guochao

摘 要 网络空间安全是国家安全的重要基石，安全测试研究通过模拟真实攻击来主动发现系统漏洞，以攻促防，对安全测试技术的研究能够有效提升防御体系的实战能力和韧性。为研究安全测试中的系统功能利用、敏感设备利用等行为的隐藏技术，文章针对系统控件进行研究，深入研究了系统消息机制和关键系统消息参数，实现了控件控制、消息控制、消息过滤等多种系统控件隐藏技术，分别对其隐藏效果和限制条件进行了实验研究，分析了各个技术的优缺点，给出了对应的防护意见和未来的改进思路。

关键词 安全测试；控件隐藏；主机安全；消息机制

doi：10.3969/j.issn.1672-9528.2026.07.042

0 引言

目前，网络空间安全已成为国家安全的重要组成部分，网络攻防对抗日趋白热化，关键基础设施、政企核心业务系统等面临的网络威胁持续升级，呈现出组织化、隐蔽化、长期化的特征[1]。为提升主动防御能力，以攻促防的安全理念得到广泛认可，通过安全测试模拟攻击者技术[2]，检验防御体系的有效性，发现系统的潜在安全漏洞，已成为网络安全建设的重要环节[3-5]。因此，研究新型的攻击技术对于完善防御体系、提升应急响应能力具有重要的现实意义[6-7]。

Windows 操作系统作为企业级环境的主流平台[8]，其任务栏系统通知区域是用户感知系统状态的关键界面。当程序调用摄像头、麦克风等敏感设备时，系统会自动在通知区域创建提示控件以提示用户敏感设备正在被使用。这一设计在增强系统透明度的同时成为敏感行为的暴露点。因此，如何在不触发用户警觉的前提下隐藏此类行为，成为安全测试技术演进的关键方向之一[9-10]。

当前任务栏系统通知区域的系统控件隐藏技术主要分为两个方向，一是利用 Windows 系统的消息机制进行隐藏；二是利用 Windows 系统的控件管理机制进行隐藏。其中，利用控件管理机制的隐藏技术通过修改系统配置文件或注册表禁用特定控件来实现隐藏，需要重启系统才能生效，缺乏灵活性，难以适应动态攻防场景，易留下可审计的操作日志。因此，本文重点对利用消息机制进行控件隐藏的主机安全测试技术进行研究，针对任务栏系统通知区域实现了基于控件控制、消息控制、消息过滤的三种控件隐藏技术，阐述了三种技术的技术原理，分析了各个技术的使用效果和优缺点，并给出了对应的防护建议，对提升安全防护能力具有重要意义。

1. 北京计算机技术及应用研究所 北京 100854

1 关键技术原理

本文深入研究了 Windows 操作系统的消息机制，明确了对任务栏通知区域通知控件的添加、删除、修改等消息机理。

1.1 Windows 系统消息机制

消息机制是 Windows 操作系统中一种基于事件消息的异步通信机制，用于在操作系统、应用程序和硬件设备之间传递事件信息，是 Windows 操作系统的核心组件[10]。消息机制引入了集中式的消息队列与分发系统，为并发执行的应用程序提供了有序共享系统资源的协调框架，有效规避了资源竞争与访问冲突，让多个应用程序能够有序地共享系统资源。消息机制将系统消息与硬件事件解耦，使应用程序无需直接处理底层硬件差异，通过标准化的消息原语（如 WM_MOUSEMOVE、WM_KEYDOWN 等）与输入设备进行交互，实现了跨硬件平台的可移植性与一致性接口。消息机制支撑了事件驱动的程序执行范式，使图形用户界面应用能够从传统的顺序控制流转变为基于事件触发的响应式架构，实现适配用户操作的随机性与交互性需求。

消息机制由消息队列、消息循环、窗口过程三层架构组成。每个窗口对应一个线程，每个线程拥有一个消息队列，并运行一个消息循环，不断从队列取出消息并分发至对应的窗口过程。消息队列作为缓冲区，允许系统异步地发送事件，使应用程序按自己的节奏处理，能够有效避免阻塞。窗口过程是每个窗口类的回调函数，负责处理特定消息。系统或其他程序可以通过发送（Send）的方式直接将消息发送给窗口过程，也可以通过投递（Post）的方式将消息添加到消息队列中，等待消息循环分发给窗口过程[11]。

消息机制的常见工作流程如图 1 所示，从控件操作开始，由系统或应用程序生成消息，通过投递的方式将消息添加到消息队列，由消息循环分发给窗口过程，或通过发送的方式直接发送给窗口过程，最后由窗口过程处理消息。例如当用户点击鼠标、按下键盘、调整窗口大小时，操作系统会按照消息机制将这些事件封装成消息放入消息队列，再由窗口对应的线程取出并处理。

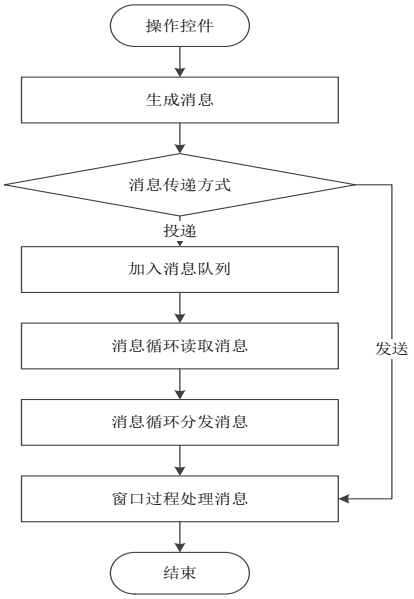

图 1 消息机制常见工作流程

消息具有固定的结构，如表 1 所示，发送给窗口的一条消息通常包括接收消息的窗口句柄、消息标识符、消息参数、消息发送时间、鼠标位置。

表 1 消息结构

```
typedef struct tagMSG {
    HWND   hwnd;      // 接收消息的窗口句柄
    UINT   message;   // 消息标识符（如 WM_PAINT, WM_KEYDOWN）
    WPARAM wParam;    // 附加参数（如按键的虚拟键码）
    LPARAM lParam;    // 附加参数（如鼠标坐标）
    DWORD  time;      // 消息发送时间
    POINT  pt;        // 鼠标位置
} MSG;
```

窗口句柄是内核对象的索引，指向系统内部维护的窗口结构，唯一标识一个窗口实例，是消息的目标地址，系统根据窗口句柄找到对应的窗口过程。窗口句柄作为消息路由的唯一标识，使得针对性进行控件控制、消息控制、消息过滤成为可能。

消息标识符是用于区分不同消息类型的整数值，唯一标识一种消息类型，决定消息语义，窗口过程根据消息标识符的值进行消息处理。常见消息标识符如表 2 所示。

表 2 消息结构

消息标识符	消息语义
WM_CREATE	初始化窗口资源
WM_PAINT	绘制界面内容
WM_SIZE	调整窗口大小
WM_COMMAND	处理用户命令
WM_NOTIFY	处理控件发送的通知
WM_TIMER	定时器到期
WM_CLOSE	用户点击关闭按钮

附加参数则需要在窗口过程进行消息处理时传递给窗口过程的数据。

1.2 通知控件消息机理

通知控件也称为托盘控件，用于显示后台运行程序的状态控件和系统通知，通常位于任务栏右下角的通知区域，包括溢出通知区域的控件和用户提示通知区域的控件。

通知控件本身不是窗口，是作为控件按钮绑定到溢出通知区域窗口或用户提示通知区域窗口的，一个窗口可绑定多个控件。通过 vs 提供的工具 spy++ 可查看到溢出通知区域窗口和用户提示通知区域窗口的具体信息，包括窗口的名称、类名和句柄，如图 2 所示。

图 2 通知控件窗口名称、窗口类名与窗口句柄

窗口句柄会随系统重启或窗口重启发生变化，但窗口类名是固定的，因此在进行控件隐藏时，需要通过窗口类名对窗口句柄进行查询，然后根据窗口句柄进行对应的操作。

在 Windows 系统中，可以通过 Shell_NotifyIcon 对窗口进行通知控件的添加、修改和删除，其消息标识符通常为

WM_USER，也可以通过 SendMessage 系列函数向窗口的通知控件发送消息，其消息标识符通常为 WM_USER+n。

通知控件的消息处理流程如图 3 所示，用户操作通知控件后，由系统根据操作事件查找控件绑定的窗口和约定的消息标识符，然后生成包含消息标识符、控件 ID、鼠标操作等信息的消息，使用投递的方式将消息添加到窗口的消息队列，目标窗口的消息循环将消息分发到对应的窗口过程，最终由该窗口过程根据消息修改控件状态。

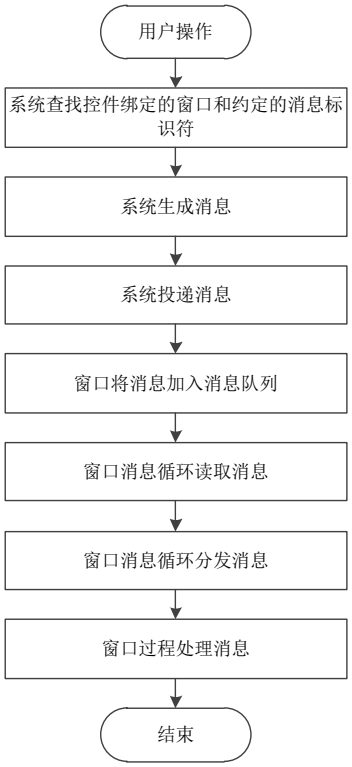

图 3 通知控件的消息处理流程

窗口通过控件 ID 和控件序号两种方式来标识控件。控件在被添加到通知区域时获取控件 ID，按照添加顺序获取控件序号。控件 ID 是通知区域用于标识控件的临时 ID，会随控件的添加和删除发生变化。控件的描述，也称为控件 TIP，一般都会有固定的标志性描述，因此可以通过控件描述来获取目标控件。

1.3 Windows 消息钩子机制

消息钩子（Message Hook）机制为应用程序提供了截获目标窗口消息的能力，通过注册回调函数，可在消息抵达窗口过程前完成捕获、篡改或丢弃，通过调用 SetWindowHook 函数来实现。这一机制的设计初衷是支持辅助功能与调试需求，也为消息过滤提供了合规的实现基础 [12]。

消息钩子机制的核心为钩子回调函数。钩子回调函数在使用时先由应用程序注册到系统中，然后在目标窗口接收消息前对这些消息进行捕获和处理。在处理时，可在钩子函数中将目标窗口的窗口过程进行替换，从而实现对消息的拦截、篡改等处理。

在实现上，通常以 dll 文件为载体，由挂钩函数、钩子回调函数、自定义窗口过程三部分组成，如图 4 所示。挂钩函数被 exe 程序调用，运行在本地进程中，负责调用 SetWindowsHookEx 将自身所在的 dll 文件注入目标窗口所在进程中。钩子回调函数在目标进程中运行，负责将原窗口过程地址修改为自定义窗口过程地址。自定义窗口过程在目标进程中运行，负责在接收窗口消息后根据需求对消息进行过滤，将放行的消息转发至原窗口过程。

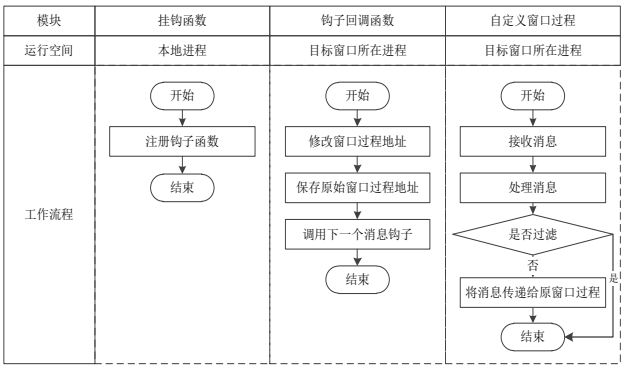

图 4 消息钩子机制三个主要模块

由于钩子回调函数会被反复调用，因此在修改窗口过程地址时需要注意不要重复修改。

2 通知控件隐藏技术

本文基于消息机制和通知控件的机理，对任务栏系统通知区域控件的删除、添加、隐藏、显示等方法进行研究，实现了基于控件控制、消息控制、消息过滤的三种控件隐藏技术。

2.1 基于控件控制的隐藏技术

基于控件控制的隐藏技术通过调用 Shell_NotifyIcon 函数控制通知区域窗口删除或隐藏目标控件，从而实现目标控件的隐藏效果。

一是需要对窗口进行遍历，获取窗口类名为 ToolbarWindow32 的窗口句柄。ToolbarWindow32 是 Windows 操作系统中用于创建工具栏控件的预定义窗口类，由 Windows 的通用控件库 Comctl32.dll 提供。

二是向窗口发送控件按钮个数获取请求，根据其控件按钮个数，逐个遍历控件按钮，根据控件描述匹配识别到目标控件，获取其控件 ID。

三是根据窗口句柄和控件 ID 构造 Shell_NotifyIcon 参数数据。Shell_NotifyIcon 参数包括消息描述符和控件数据。控件数据中包含窗口句柄、控件 ID、控件消息描述符、控件句柄等信息。在隐藏控件时，Shell_NotifyIcon 消息描述符配置

为 NIM_MODIFY，具体如表 3 所示。

表 3 Shell_NotifyIcon 参数配置

```
NOTIFYICONDATA nid;
nid.cbSize = sizeof(nid);
nid.hWnd = hWnd;
nid.uID = uID;
nid.uCallbackMessage = WM_USER;
nid.uFlags = NIF_STATE;
nid.dwState = bShow ? NIS_SHAREDICON : NIS_HIDDEN;
nid.dwStateMask = NIS_HIDDEN;
nid.hIcon = LoadIcon(GetModuleHandle(NULL),
MAKEINTRESOURCE(0));
Shell_NotifyIcon(NIM_MODIFY, &nid);
```

重新显示控件则只需将参数中的 NIS_HIDDEN 改为 NIS_SHAREDICON。

2.2 基于消息控制的隐藏技术

基于消息控制的隐藏技术通过调用 SendMessage 函数控制通知区域窗口删除或隐藏目标控件，从而实现目标控件的隐藏效果。

首先，根据窗口类名获取窗口句柄。然后，向窗口发送控件按钮个数获取请求，根据其控件按钮个数，逐个遍历控件按钮，根据控件描述匹配识别到目标控件，获取其在控件列表中的序号和控件 ID。

获取窗口句柄和控件序号后，调用 SendMessage 函数以图表序号为参数向窗口发送对应的消息标识符。

在删除控件时，使用消息标识符 TB_DELETEBUTTON，使用控件序号标识控件。在隐藏控件时，使用消息标识符 TB_HIDEBUTTON，使用控件 ID 标识控件。SendMessage 函数调用代码如表 4 所示。

表 4 SendMessage 调用代码

功能	代码
删除	SendMessageA(hWnd, TB_DELETEBUTTON, index, 0);
隐藏	SendMessageA(hWnd, TB_HIDEBUTTON, ID, 1);

由于删除控件后，会在通知区域留下一个空白位置，需要对通知窗口进行刷新操作。Windows 窗口有等级划分，高级窗口自动过滤低级窗口的消息，所以给窗口发送更新消息是无效的。通知窗口在调用 Shell_NotifyIcon 函数添加或删除按钮时会自动刷新，因此可以利用 Shell_NotifyIcon 函数来刷新通知窗口。

重新显示控件需要重启桌面；重新显示隐藏的控件则需要将参数 1 改为 0。

2.3 基于消息过滤的隐藏技术

基于消息过滤的隐藏技术通过调用 SetWindowHook 函数设置消息钩子，在钩子函数中修改窗口过程函数，在自定义的窗口过程函数中拦截目标按钮的添加消息，从而实现隐藏的效果。

系统通过 Shell_NotifyIconW 函数添加通知控件。经逆向分析、动态调试 Shell_NotifyIconW 函数可知添加通知控件时系统或应用程序会向 Shell_TrayWnd 窗口发送 WM_COPYDATA 消息。经查询 MSDN，发送 WM_COPYDATA 消息时，SendMessage 的参数 lParam 类型为 COPYDATASTRUCT，其中的 lpData 成员是要传递的数据。在添加通知控件时，lpData 成员存储的就是 ICONINFOEXW。删除和隐藏时则不一定。

基于消息过滤的隐藏技术需要在 dll 文件中实现用于钩取窗口消息的钩子回调函数、用于过滤消息的自定义窗口过程和用于注册钩子的挂钩函数。

挂钩函数执行后，完成钩子回调函数注册，系统会将 dll 文件注入目标进程中运行。钩子回调函数为避免重复修改原窗口过程，会调用 UnSetWindowHookEx 注销本钩子函数，然后通过调用 SetWindowLongPtr 函数设置窗口的 GWL_WNDPROC 属性，实现窗口过程的地址替换，如果替换失败则卸载自身模块。

自定义窗口过程要过滤的目标信息是系统发送给托盘窗口的用于创建托盘控件的信息。自定义窗口过程通过提示文本识别目标消息。在创建通知控件时，系统会在控件信息结构 ICONINFOEXW 中的字段 szModName 存储控件的提示文本。当自定义窗口检测到消息类型为 WM_COPYDATA 时，检查消息对应的数据，如果数据的提示文本与目标控件的提示文本相同则不予传递，如果不同则将消息传递给原窗口过程。消息过滤的核心代码如表 5 所示。

表 5 消息过滤的核心代码

```
switch (uMsg){
  case WM_COPYDATA:
      COPYDATASTRUCT* pCD = (COPYDATASTRUCT*)
lParam;
      pCD->lpData;// 这里存储 ICONINFOEX
    ICONINFOEXW* picon = (ICONINFOEXW*)pCD->lpData;
      if (wcsstr(picon->szModName, L " 麦克风 ")) // 拦截
        return 0;
}
CallWindowProc((WNDPROC)OldProc, hwnd, uMsg, wParam,
lParam);
```

如果控件在进行消息挂钩之前完成创建，无法对其创建消息进行挂钩，即无法实现隐藏。对于已经完成创建的控件，可以通过重启桌面的方式使其重新被创建，通过对新建控件的过滤实现隐藏。

3 可行性验证

为验证本文所研究的通知控件隐藏技术的可行性，在 Windows 10 环境中分别对系统通知控件、应用程序通知控件进行隐藏可行性试验。

试验所选系统控件为蓝牙设备控件，所选应用程序控件为 Snipaste 截图软件控件。蓝牙控件是常驻系统通知控件，与 Wi-Fi、电池、音量等控件同级，显示在 Windows 任务栏托盘区用于快速开关蓝牙、查看连接状态、配对 / 管理设备，与 Wi-Fi、电池控件等系统控件具有相同特性。Snipaste 截图软件是一款普通应用软件，在任务栏中注册了一个通知控件，具有普通应用软件通知控件的普遍特性。

在实验环境中，蓝牙设备通知控件的消息描述符为“蓝牙设备”，Snipaste 截图软件通知控件的消息描述符包含“Snipaste”。

3.1 基于控件控制的隐藏技术

通过 Shell_NotifyIcon 函数分别在管理员权限和非管理员权限下进行隐藏，结果如表 6 所示。

表 6 基于控件控制的隐藏技术实验结果

控件	管理员权限	非管理员权限
蓝牙设备控件	失败	失败
Snipaste 截图软件控件	成功	成功

Snipaste 截图软件控件隐藏效果如图 5~6 所示。

图 5 隐藏前（右二）

图 6 隐藏后

基于控件控制的隐藏技术以 Shell_NotifyIcon 函数为核心，操作目标是控件，优点是通知控件隐藏后不会留有痕迹，缺点是只能隐藏普通应用程序的通知控件，无法隐藏系统通知控件。

3.2 基于消息控制的隐藏技术

通过 SendMessage 函数分别在管理员权限和非管理员权限下进行隐藏，结果如表 7 所示。

表 7 基于消息控制的隐藏技术实验结果

控件	管理员权限	非管理员权限
蓝牙设备控件	成功	成功
Snipaste 截图软件控件	成功	成功

蓝牙设备控件隐藏效果如图 7~8 所示，Snipaste 截图软件控件隐藏效果与蓝牙设备控件相同。

图 7 隐藏前（左二）

图 8 隐藏后

基于消息控制的隐藏技术以 SendMessage 函数为核心，操作目标是按钮，优势是能够隐藏系统通知控件，劣势是控件隐藏后会留有明显空白。

通过 SendMessage 函数删除按钮并通过 Shell_NotifyIcon 函数强制窗口更新，则可以解决出现空白的问题，效果如图 9~10 所示。

图 9 操作前（左二）

图 10 操作后

3.3 基于消息过滤的隐藏技术

通过消息过滤技术分别在管理员权限和非管理员权限下进行隐藏，结果如表 8 所示。由于蓝牙设备控件和 Snipaste 截图软件控件在进行消息挂钩之前已完成创建，无法对其创建消息进行挂钩，也就无法实现隐藏。本文在完成消息挂钩后，对桌面进行重启，过滤其新建控件消息，实现控件隐藏。

表 8 基于消息过滤的隐藏技术实验结果

控件	管理员权限	非管理员权限
蓝牙设备控件	成功	成功
Snipaste 截图软件控件	成功	成功

蓝牙设备控件隐藏效果如图 11~12 所示，Snipaste 截图软件控件隐藏效果与蓝牙设备控件相同。

图 11 隐藏前（左二）

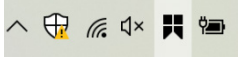

图 12 隐藏后

基于消息过滤的隐藏技术以消息钩子为核心，优势是能够隐藏系统通知控件，劣势是需要重启桌面，容易被察觉。

4 结语

本文聚焦安全测试演进的研究需求，为隐藏红队的系统功能、敏感设备等利用行为，对 Windows 系统任务栏通知区域的通知控件隐藏技术进行研究，明确了 Windows 系统的消息机制、通知控件的消息机理和 Windows 消息钩子机制，在此基础上实现了控件控制、消息控制和消息过滤的三种通知控件隐藏技术，分别在管理员权限和非管理员权限下对系统通知控件隐藏效果和普通应用程序通知控件隐藏效果进行了充分实验。实验表明，基于控件控制的隐藏技术隐藏控件后不会留有痕迹，但无法对系统通知控件进行隐藏。基于消息控制的隐藏技术能够对系统通知控件进行隐藏，但在隐藏后会留有空白，痕迹明显，需要与窗口更新功能相结合解决隐藏痕迹的问题。基于消息过滤的隐藏技术能够对系统通知控件进行隐藏，但只能对新建控件进行隐藏，无法隐藏已完成创建的控件，需要与桌面重启功能相结合进行解决。三种方法各有优缺点，在实际使用中，需要根据使用场景进行选择。

在防护方面，一方面可针对技术所用核心函数进行监控，另一方面可针对技术所特有的行为特征进行监控。对核心函数进行监控时，通过函数参数和调用栈进行精准识别，对可疑进程的调用行为进行拦截，对普通进程的调用行为进行告警。基于控件控制的隐藏技术和基于消息控制的隐藏技术均需要进行远程进程读取，基于消息过滤的隐藏技术需要进行消息挂钩，因此，可对远程进程读取合并核心函数调用的行为、普通程序的消息挂钩行为进行重点监控。

在下一步研究中可与函数隐藏调用技术、远程线程隐蔽注入技术等结合，实现核心函数调用行为的隐藏，实现远程进程读取等特征行为的隐藏。因此，在防护方面，也需要加强对函数隐藏调用、远程线程隐蔽注入等技术的监控。

参考文献：

[1] 桂畅旎 , 徐国庆 .2025 年国际网络安全态势回顾与趋势展望 [J]. 中国信息安全 ,2026(1):25-29.

[2] 王保魁 , 吴琳 , 贺筱媛 , 等 . 美军红队评估体系建设现状及启示 [J]. 国防科技 ,2020,41(2):91-95.

[3] 吉梁 , 张士莹 , 王逸鹤 . 网络攻防技战法及对策建议浅析 [J]. 网络安全和信息化 ,2023(4):38-41.

[4] 王乾 . 基于动态二进制分析的关键函数定位技术研究 [D]. 郑州 : 解放军信息工程大学 ,2012.

[5] 朱贤斌 , 程海翔 . 探讨网络安全应急演练和实战攻防演习 [J]. 网络安全和信息化 ,2021(4):133-141.

[6] 李欣 , 高见 . 网络攻防演练及其现实意义 [J]. 中国信息安全 , 2019(1): 33-35.

[7] 袁誉峰 , 韩保礼 , 张晓峰 , 等 . 计算机网络信息安全及防护策略研究 [J]. 中国新通信 ,2024,26(8):7-9.

[8] 龙伟 . 基于动态追踪和进程克隆的 Windows 模糊测试技术研究 [D]. 成都 : 电子科技大学 ,2024.

[9] 程曦 . Windows10 环境下的程序隐藏技术研究 [D]. 哈尔滨 : 哈尔滨工业大学 ,2024.

[10] 梅亦 . 基于实战的网络安全攻防演练探索 [J]. 数字通信世界 , 2025(8):169-171.

[11] 朱方祥 , 顾乃杰 . 一种 Windows 应用程序的 GUI 自动化遍历方法 [J]. 计算机应用与软件 ,2018,35(11):26-32.

[12] 赵志恒 , 于秀山 , 黄松 , 等 . 基于 Windows Hook 的 GUI 测试操作捕获方法 [J]. 计算机工程与设计 ,2016,37(3):660-664.

【作者简介】

任益辰（1995—），女，河北衡水人，硕士，工程师，研究方向：主机安全。

汪美琴（1996—），女，湖北黄冈人，硕士，工程师，研究方向：智能安全。

刘明哲（1996—），男，北京人，硕士，工程师，研究方向：系统安全。

董国超（1995—），男，北京人，硕士，工程师，研究方向：Web 安全。

（收稿日期：2026-06-16　修回日期：2026-07-15）