

Automated Blocking of Malicious Code with NDIS Intermediate Driver

Lee Ling Chuan*, Chan Lee Yee**, Mahamod Ismail**, Kasmiran Jumari**

*Malaysia Computer Emergency Response Team, CyberSecurity, Malaysia

**Department of Electrical, Electronic and System Engineering, Universiti Kebangsaan Malaysia

lc.lee@cybersecurity.my, chanleeyee@gmail.com, mahamod@vlsi.eng.ukm.my, kbj@vlsi.eng.ukm.my

Abstract— With the evolution of malware technology, modern malware often hide its malicious behaviour in various methods. One of the popular manners is to conceal the network communication. This concealment technique poses obstacles to security mechanisms, which detecting the malicious behaviours. In this paper, we give an overview of the automated blocking malicious code project, a new approach to computer security via malicious software analysis and automatic blocking software. In particular, this project focuses on building a unified executable program analysis platform and using it to provide novel solutions to a broad spectrum of different security problems. We propose a technique for the Network Driver Interface Specification (NDIS) integrate together with a unified malicious software analysis platform. The NDIS model supports hybrid network transport NDIS drivers, called NDIS intermediate drivers. This driver lies between transport driver and NDIS driver. The advantage of using NDIS intermediate drivers is, it can see the entire network traffic taking place on a system as the drivers lie between protocol drivers and network drivers. By intercepting security-related properties from network traffic directly, our project enables a principled, root cause based approach to computer security, offering novel and effective solutions.

Keywords— Network Driver Interface Specification, NDIS Intermediate Driver, Interception, Malicious Traffic, Malware Analysis

I. INTRODUCTION

Over the past year, the number of programs developed for malicious and illegal purpose has grown rapidly. Malicious code has been categorized neatly (e.g. viruses, worm or Trojan Horses) based upon functionality and attack vector. Indeed, criminals are making extensive use of malware to control computers and steal personal, confidential, or otherwise proprietary information for either profit or for fun. The evolution of modern malware now a day is not only attack inside computer system, but through the convenience of the Internet, spreading from one victim computer to another. There are also some malicious codes, conceal the communication pathway, and avoid the detections from security protection mechanisms such as firewall, sniffer, antivirus, IDS system etc.

In general, security mechanisms on Windows such as firewall, Intrusion Detection System (IDS), etc. rely on the

native TCP/IP for network traffic related functions. However Microsoft has imposed restrictions on raw socket as below:

- TCP data cannot be sent over a raw socket.
- UDP datagram cannot spoof their source address over a raw socket
- Raw sockets cannot make calls to the bind() function.

TABLE 1. COMPARISON WITH NDIS AND WSK AND WINSOCK [5]

Interface	Benefits	Drawbacks
Winsock	▪ Easy to use, well documented	▪ Easier to track down
WSK	▪ Uses the existing TCP/IP stack ▪ Not as easy to track down	▪ More demanding and less forgiving than Winsock ▪ Must account for protocol-dependent behaviour
NDIS	▪ Offers the most control ▪ Can spoof packets ▪ Can bypass local firewall	▪ Effort required to implement a new TCP/IP stack ▪ Switches may limit one MAC address per port ▪ Can be conspicuous in packet capture

A raw socket is a socket that allows direct access to the headers of a network frame. Naturally, the freedom to spoof frame information was abused by malware developers. Hence, these restrictions have been imposed by Microsoft on Windows XP SP2 and later version. The constraints placed on raw sockets are built into tcpip.sys and tcpip6.sys drivers [5]. Only one solution that can circumvent the restrictions that places on raw sockets is to roll a dedicated transport layer. This approach gives the authority to control over the packets that created. Thus, to see the entire network traffic taking place on a system, rolling NDIS protocol driver is the only solution on Windows, especially Windows XP SP2, Microsoft Vista or Windows 7 [2]. Table 1 shows the comparison between NDIS interface with WSK and Winsock Interface.

In this paper we address a design system for intercepting and automating blocking of malicious network traffic by using NDIS Intermediate driver method [3] [4]. The reason that we choose NDIS intermediate driver as our automated project is NDIS can capture all the packets passing through the system, including packets such as rootkit that can bypass local firewall [9]. Our approach is to capture the entire network traffic

packet and control it in a real time manner. An output of the network traffic log file will export into a user readable file, and can display any field in any headers protocol as user want.

II. BACKGROUND AND RELATED WORK

Host based network security software on Windows have made significant advances in both technology and scope of deployment within the past few years. Despite these advances, many challenges are remain. One of the biggest challenge is they are mostly relying on the underlying Operating System's support for data gathering and monitoring. However, the evolutions of malware programs have proof that they are capable to exploit the weakness. The capability to bypassing virtually all commodity, host-based firewall and intrusion detection system software on the market today has force security researchers seeking new method on detecting and blocking the malicious activities, and there is security organizations decided to use NDIS Intermediate Driver as the solution.

Kaspersky Anti-Virus is one of the security products on the market which implemented the technology of NDIS Intermediate Driver. Although Kaspersky Anti-Virus software allows their users to disable the kaspersky Anti-Virus NDIS Filter function. However, disabling the function will cause the Anti-Hacker/Firewall function not to perform packet filtering and detecting network attacks functions as the network packets will not be intercepted [10].

According to [1], the current firewall technology on defending against external network attacks and threats which deals with the packets under user mode has a lot of limitation. It has proposed that the protection can be done better by using NDIS intermediate driver.

III. NDIS_PACKET REPRESENTATION

The main function of NDIS packets (represented by NDIS_PACKET structure) is to ensure all network data can be sent to or from the network in a system. Prior sending data on the network, a protocol driver allocates NDIS packets, (represented by a NDIS_PACKET), filled with data, and passed to the next lower NDIS driver. On the contrary, some lowest level NIC drivers will allocate packets to hold received data and pass it to interested higher layer drivers. There are functions provided by NDIS for allocating and manipulating the substructures that form a packet.

In general, each NDIS Packet is a Packet Descriptor, and it has a series of Buffer Descriptor or NDIS_BUFFER [7]. The number of chained for NDIS_BUFFER is depending on the size of the NDIS packet. If the packet size is small, there will be only one chained for NDIS_BUFFER and this buffer enough to describe the range of virtual memory that contains the complete packet data. Figure 1 shows that the NDIS_PACKET which has two chained of NDIS_BUFFER. The first NDIS_BUFFER describes the range of virtual memory that contains the Ethernet header while the second NDIS_BUFFER describe the range of virtual memory that contains the Ethernet payload [8].

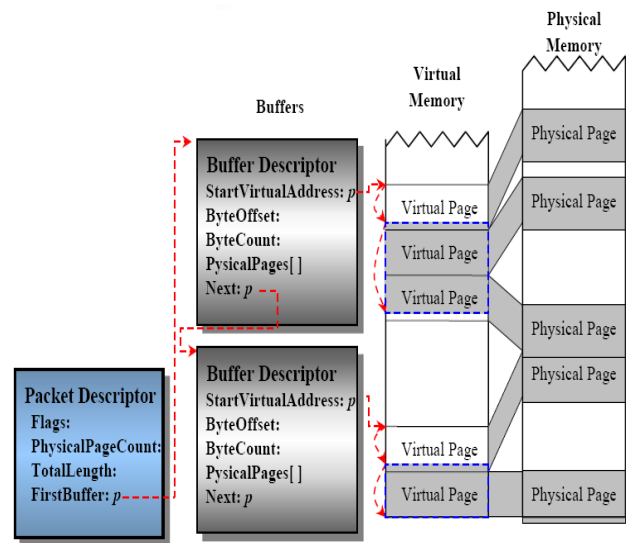


Figure 1. Multi-Buffer NDIS_PACKET Illustration [6]

The content of a typical packet descriptor are as the following list:

- Private areas for the miniport NIC driver and a protocol driver
- Flags associated with the packet, defined by a cooperating miniport(s) and protocol driver(s)
- Number of physical pages that contain the packet
- Total length of the packet.
- Pointer to the first buffer descriptor that maps the first buffer in the packet.

The following list shows the contents of a typical buffer descriptor:

- Starting virtual address of each buffer
- Buffer's byte offset into the page pointed to by the virtual address
- Total number of bytes in the buffer
- Pointer to the next buffer descriptor, if any
- Virtual range, possibly spanning more than one page that makes up the buffer described by the buffer descriptor. These virtual pages map to physical memory.

The virtual range allocates the buffer described by the buffer descriptor. These virtual pages will map to physical memory.

IV. INTERCEPTION AND BLOCKING ARCHITECTURE

The NDIS model supports hybrid network transport NDIS drivers, called NDIS intermediate driver. The driver is lie between transport drivers and NDIS drivers. To an NDIS driver, an NDIS intermediate driver looks like a transport driver; to a transport driver, an NDIS intermediate driver looks like an NDIS driver.

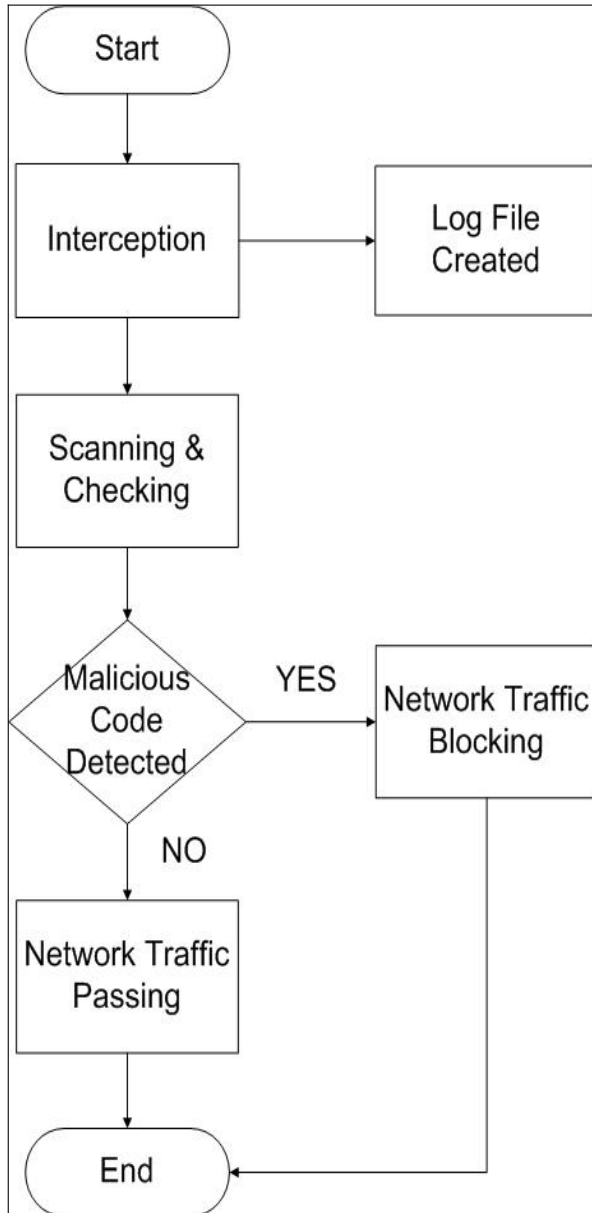


Figure 2. Interception and Blocking Architecture

In this section, we give an overview of the interception and blocking architecture using NDIS intermediate driver. Figure 2 shows the flow chart for the interception and blocking that have been developed.

As shown in Figure 2, all network traffics are first intercepted by the interception mechanism. In the process of interception, the entire network packets will dump into a log file in hexadecimal format. The purpose of creating a log file is to see what activities is taking place. The scanning and checking mechanism will execute pattern matching function. This function will compare the intercepted network traffic with pre-defined unique signature strings where each unique

string represents malicious portion code signatures. Knuth-Morris-Pratt (KMP) algorithm [11] [12] was used to observe when a mismatch occurs. The KMP algorithm looks for the malicious code signature pattern in a left-to-right order. It looks like the brute force algorithm but it shifts the pattern more intelligently than the brute force algorithm. Figure 3 illustrates the fragment code of this function. If the pattern matching occurs, NDIS intermediate driver will immediately drop the interception network traffic. STATUS_DROP function can be used to drop the network packet as shown in Figure 4. If none of the pattern is match, network traffic will allow pass through on the system.

```

TT[0] = -1;
TT[1]=0;
l=2;
k=0;

while(T[l] !='\0') {
    if (T[l-1]==T[k]) {
        TT[l] = k+1;
        k++;
        l++;
    }
    else if (k > 0) {
        k=TT[k];
    }
    else {
        TT[l] = 0;
        l++;
        k=0;
    }
}

if(pPacketContent[34]==0 && pPacketContent[35]==80)
{
    DbgPrint("Receive Http Port 80");
    for(i=54;i<=DataOffset;i++)
    {
        while(pPacketContent[i+j]!='\0' && T[j]!='\0')
        {
            if (pPacketContent[i+j] == T[j])
            {
                DbgPrint("Signature Match Success");
                ++j;
                if(T[j]=='\0')
                {
                    DbgPrint("myndis blocking - malicious traffic detected");
                    return STATUS_DROP;
                }
            }
            else
            {
                i += j - TT[j];
                if (j > 0)
                    j += TT[j];
            }
        }
    }
}
}

```

Figure 3. Knuth-Morris-Pratt (KMP) Pattern Matching Algorithm

```

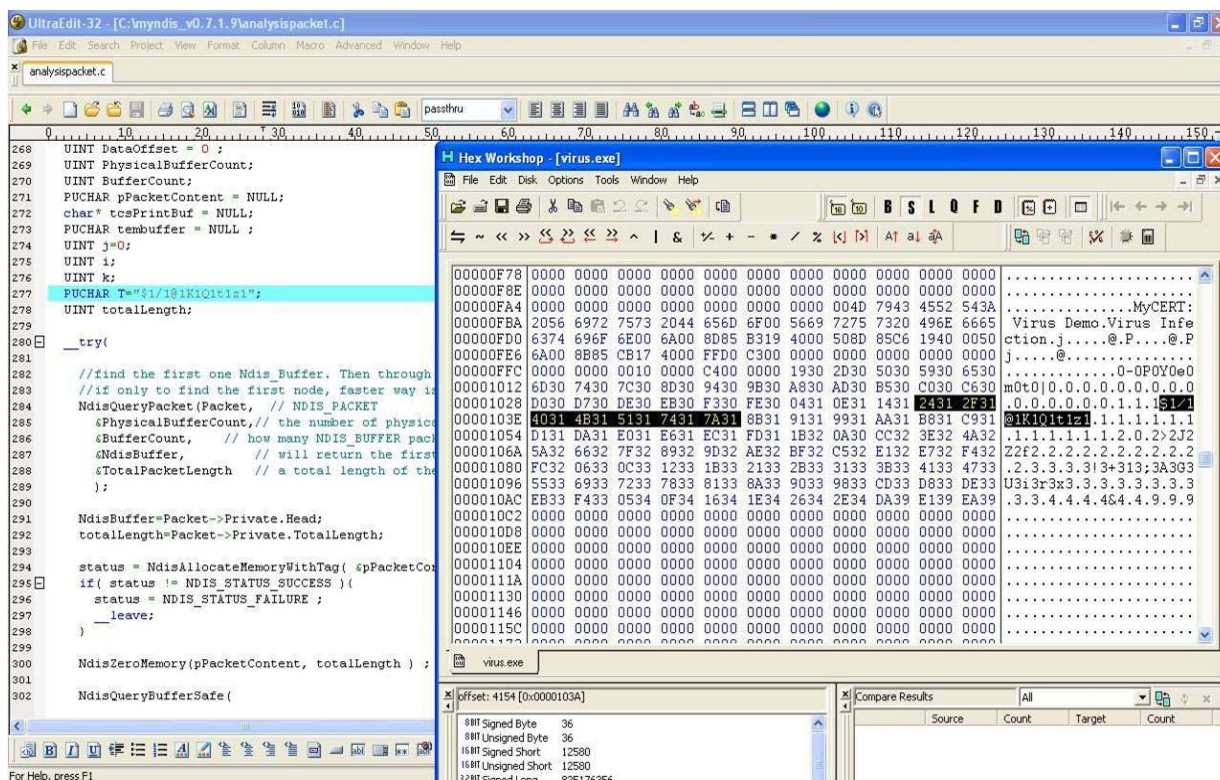
DbgPrint("Signature Match Success");
++j;
if(T[j]=='\0')
{
    DbgPrint("myndis blocking - malicious traffic detected");
    return STATUS_DROP;
}

```

Figure 4. Blocking Function Based on Signature

V. CASE STUDY

We implemented a system for malicious network traffic interception and analysis with the above component and technique. In this section, an Adobe Flash Player exploit



[13][14] will be exploited in a testing machine. All incoming and outgoing network traffic will be intercepted by NDIS intermediate driver. The signature of the Adobe Flash Player exploit will be collected and put under the list of signature matching. Figure 5 shows the unique string for Adobe Flash Player exploit payload.

As discussed in section VI, all intercepted network traffic will be dumped into a hexadecimal log file. Figure 6 shows the example of hexadecimal log file that match with the malicious payload signature.

TIME: 2010-07-19 10:55:37

PROTOCOL : 06

SRC_IP_ADDR:203.223.144.18

DEST_IP_ADDR:192.168.117.125

SRC_PORT:0080

DEST_PORT:0439

PAYLOAD:

```

0 e8 c6 ea ff ff 8b 45 fc 8b 5d 8 89 30 8b 3 8b 0 ff 70 4 e8 ec ea ff ff 8b 4d fc 89 1
3 f3 a4 8b 45 f8 ff 70 4 e8 68 ea ff ff 8b 45 f8 83 60 4 0 8b 3 8b 40 4 ff 70 4 e8 8
5 fc ff 8b 8 18 e8 ff ff 8b 45 fc 89 30 a1 ac 4f 1 1 8b 0 ff 70 4 e8 3e e8 ff ff 8b 4d
ff ff 30 83 60 4 0 8b 75 c 8b 46 4 ff 70 4 e8 ef e6 ff ff 8b b 89 41 4 8b 76 4 8

```

Figure 6. Hexadecimal Log File in Local PC

After executed NDIS Intermediate driver, all network packets passing through the testing machine will be intercepted. Pattern matching function will execute to scan the

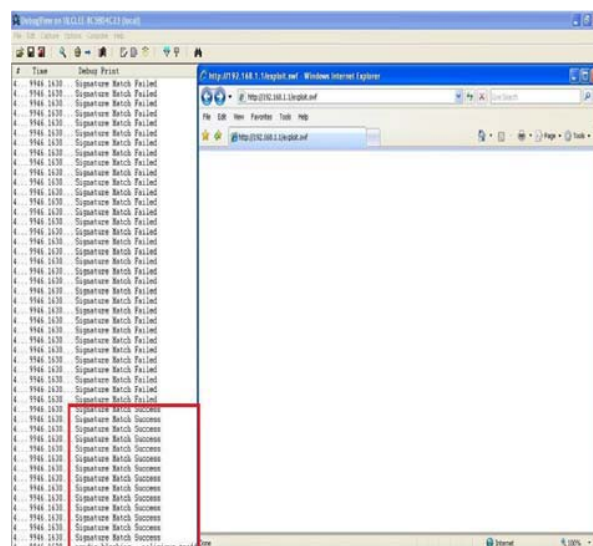
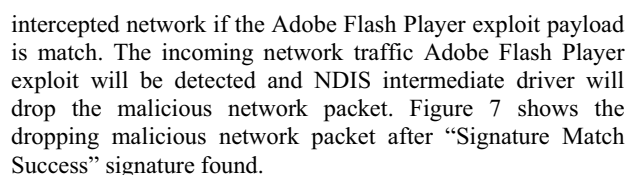


Figure 7. Blocking Malicious Network Traffic Packets

VI. CONCLUSIONS

Malware scanning engine is the core technology of today malware protection and the first line of defence against external network attacks and system threats. Most personal scanning engine deals with the network packets under the user mode, however there are a lot of limitation. In order to provide better protection for user, security prevention mechanisms need to be done at kernel mode. This paper clearly describes the use of NDIS intermediate driver and the development of scanning engine model at kernel model.

REFERENCES

- [1] He Chaokai, "Design and implementation of a personal firewall Based on NDIS Intermediate Drivers," IEEE Eighth ACIS International Conference, Qingdao, Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 878-882, July30 2007 – August 1 2007.
- [2] Muna M. Taher Jawhar, Monica Mehrotra, "System Design for Packet Sniffer using NDIS Hooking," International Journal of Computer Science & Communication, Vol. 1, No. 1, pp. 171-173, January – June 2010.
- [3] Walter Oney., "Programming the Microsoft Windows Driver Model", 2nd ed. Microsoft Press, A Division of Microsoft Corporation 2003.
- [4] Mark E. Russinovich, David A. Solomon, Alex Ionescu, Windows® Internals: Including Windows Server 2008 and Windows Vista, Fifth Edition, Microsoft Press, A Division of Microsoft Corporation, One Microsoft Way, 2009, pp. 1053 - 1071.
- [5] Reverend Bill Blunden, "The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System," Wordware Publishing Inc, 2009, pp. 610-616.
- [6] HanTechSnS, "SecurityStudio 2001: NDIS Intermediate Driver Operation & WDM Driver Operation," pp. 6, 2001
- [7] Printing Communications Associates, Inc. (PCAUSA), "NDIS_PACKET Discussion Part 2: NDIS_PACKET Reserved Areas," January 17, 2010.
- [8] MSDN Library, Microsoft Corporation, "NDIS-Supplied Packet and Buffer Handling Functions (NDIS 5.1)," March 6, 2010.
- [9] Greg Hoglund, James Butler, "Rootkits: Subverting the Windows Kernel," Addison Wesley Professional, July 22, 2005, Chapter 9: Covert Channels.
- [10] (2010) The Kaspersky website. [Online]. Available: <http://support.kaspersky.com/faq/?qid=208279317>
- [11] Greg Plaxton, "String Matching: Knuth-Morris-Pratt Algorithm," Theory in Programming Practise, Department of Computer Science, University of Texas at Austin, Spring 2004.
- [12] J. H. Morris, Jr and V. R. Pratt. "A linear pattern-matching algorithm," Report 40, University of California, Berkeley, 1970.
- [13] Mark Dowd, "Application-Specific Attacks: Leveraging the ActionScript Virtual Machine," IBM Internet Security Systems, April 2008. [Online]. Available: http://documents.iss.net/whitepapers/IBM_X-Force_WP_final.pdf
- [14] (2010) Adobe Systems Incorporated, "Flash Player update available to address security vulnerabilities," APSB08-11, April 8, 2008. [Online]. Available: <http://www.adobe.com/support/security/bulletins/apsb08-11.html>