Remote Code Execution using ICMP Modified Structured Storage Covert Channels Without Elevation of Privileges

Aditya T*
*School of Electronics and Communication Engineering
PESIT, Outer Ring Rd, Banashankari 3rd Stage, Banashankari, Bengaluru, Karnataka 560085, India
Email: adityat@iisc.ac.in

***Abstract*—This paper discusses the vulnerability of modern anti-malware softwares to covert channel attacks without the need for elevation of privileges by employing an inherent vulnerability of ICMP and Winpcap. A modified form of *Network Storage Covert Channels* is introduced which is capable of evading the present detection algorithms based on statistical and information-theoretic models. The vulnerability pertaining to bypassing administrator privileges creates a special cause for concern since with the recent emergence of *Ransomware*, the only line of defence that most novice users usually have is the spurious request for administrator privileges upon executing a malware masquerading as a harmless file but with this bypassed the threat of *Ransomware* extends to even proficient users. Most users if not all of wireshark and other network virtualization software such as GNS3 depending on Winpcap, select the option to automatically start *NPF.sys* driver at Windows boot. This default setting is exploited to present an ideal environment for establishing covert channels.**

***Index Terms*—Covert Channels, Remote Code Execution, ICMP Exploit**

I. INTRODUCTION

Today's world being composed of ever so intricately connected sub-systems, some responsibility falls on the user to use his fair judgment when on the internet. Typically, the ignorance on the part of the user is exploited by malicious software to carry out their ploy. Most notable being the recent emergence of notorious ransomwares such as crypto locker and so on, which require the users to grant them administrator access. However, we see in this paper, the existence of some rather useful softwares can inadvertently bypass the need to acquire elevation of privileges which raises a concern since malicious software can now depend on these to attack even proficient users.

The idea behind covert channels was first conceived by Lampson.[1973] while studying the problem of confinement. The problem of confinement is major security issue when MLS(Multi Level Secure) systems are concerned (Serdar 2006 ; Son and Foss 2006) where different users are restricted to different access permissions but some programs posses the sufficient authorization to transcend across the user space. This allows those programs to either deliberately or accidentally leak information belonging to a higher access space to those of lower access space. Regarding this, Lampson studied policies of complete "isolation" of user spaces (Virtual Machines) wherein no program can transcend spaces and "transitivity" a policy intermediate to complete "isolation" and "freedom" where services dependent on confined programs(Schaefer *et al*. 1977) are in-turn confined to prevent possible leaks. But despite measures proposed by Lampson to eliminate covert channels, presence of certain applications and default choices imposed by those applications allow programs to cross user spaces with different access specifications thus creating an ideal environment for establishing covert channels. In this paper, we present one such environment created by the existence of Winpcap and a certain default choice chosen during its installation which facilitates the creation of covert channels and also remote code execution.

A. Background

ICMP vulnerabilities have been familiar with the internet community for a fairly long time. Worms like the Welchia Worm have traditionally used ICMP pings to map the network before targeting their victims (Staniford,Paxson and Weaver 2002 ; Symantec 2007) and one can even attribute the default no response to pings on Windows boxes as a consequence of such attacks. This was followed by introduction of covert channels using ICMP which was extensively documented by (alhambra and daemon9 1996 ; Fisk 2002), the *spoofed host* method discussed in (alhambra and daemon9 1996) is impractical without a robust covert channel to go along with. We present a robust formulation for storage based covert channels which allows the *spoofed host* method to work in the current network architecture.

Covert channels in general are defined as *a channel*

or medium of communication that exist between entities by bypassing any firewall/security feature which would otherwise have thwarted the existence of such a medium. Traditionally, covert channel consists of a shared resource which acts as a medium for communication and an event which is used to modulate the shared resource with the information to be sent. Covert channels proved to be quite an effective means of tunneling malware into host systems. The first of the covert channels proposed were primarily present in MLS(Multi Level Secure) systems(Karger and Wray 1991) which included file locks,bus channels etc being used as shared resource(Patrick and Gallagher 1993). These were followed by implementations in transport layer architecture(Giffin *et al*. 2002 ; Song and Ephremides 2005). But despite having different shared resources they typically depend on an event synchronized amongst two or more users engaging in the covert channel to manipulate the shared resource hence allowing the other users to interpret the information modulated by the event. This clearly presents a flaw since these types of covert channels can be identified by the occurrence of such timed synchronized events rather than through the information/shared resource they depend on. (Serdar 2006 ; Ahsan 2002) propose methods of detection for such synchronized event based covert channels and have published results which have detected the existence of these channels with fair amount of accuracy.With the usage of anti-malware systems becoming more and more prevalent their ferocity declined precisely due to their repetitive behaviour which allowed anti-malware systems to identify their existence. These were subsequently followed by DOS and distributed DOS using smurf attacks which once again declined with servers becoming more intelligent and hence weeding out such requests. In this paper, we discuss and demonstrate how despite such effective preventive measures, covert channels can still be setup and used for remote code execution without the need for elevation of privileges due to existence of vulnerabilities in Winpcap, Wireshark, GNS3 etc. We also formulate relation between throughput and detection time for binary modulation scheme covert channels.

II. Storage and Timing Network Covert Channels

One class of covert channels are *Network Covert Channels* which use links in the network as a shared resource. A comprehensive survey of various types of network covert channels are documented in (Wendzel et al. 2015 ; Couture 2010). These are again broadly classified into two types, *Storage Covert Channels* and *Timing Covert Channels* although some (Wray 1991) have argued that covert channels cannot be clearly distinguished as either and are rather a hybrid of both but in this paper we follow the definition proposed by Kemmerer.[1983] with a slight change to *Timing Covert Channels* and Wendzel et al.[2015:2]. Kemmerer's definition holds for in-system covert channels where the users involved share

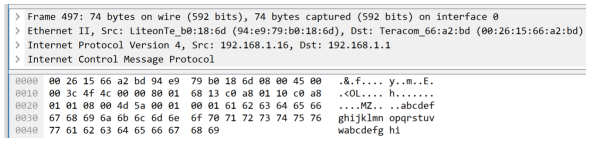


Fig. 1. Default ICMP Payload on Windows box

the system but are differentiated based on their access space, in such a scenario, sharing a reference clock is possible but in this paper we deal with *Network Covert Channels* where the users are based off of entirely different systems hence sharing a reference clock is not realistically possible(More interested readers can look at *preambles* which are typically the first few bits of a sequence used for clock synchronization, while they can work they also increase regularity hence compromising the channel). The *Timing Covert Channels* use the packet inter-arrival time as the synchronized event for the other users to interpret the modulated information and *Storage Covert Channels* use fields in the packet header to alter the header length which is then interpreted by the receiver. Fisk et al.[2002] introduced two types of *Storage Covert Channels* based on steganographic means of hiding data. Those that used header fields were classified as structured and those that used the payload/data field were classified as unstructured. But, since both *Storage and Timing Covert Channels* depend on multiple packets to transfer a symbol(combination of bits) this implies that certain regularity in the packet arrival is to be expected in-order to ensure correct interpretation and hence can be easily detected using information-theoretic methods.(Serdar,Brodley and Shields 2008) published results that detected event based covert channels with over 95% accuracy.

The drawback of traditional event based covert channels(Storage and Timing) is the use of multiple packets to transfer a single symbol. This implies that the reconstruction of the symbol at the receive end requires the packets to adhere to a sequence and a time interval limit beyond which the symbol cannot be reconstructed. This adherence causes regularity hence compromising the covert channel. The unstructured storage covert channel alternative does posses a significant size to transmit data but can be compromised by comparing it with the default payload(Figure 1). Hence, we propose a modified form of structured *Storage Covert Channel* using this we counter the claims of storage covert channels being more repetitive than timing covert channels. By transmitting one or more symbols as opposed to bits per packet removes the need for the packets to arrive within any specific time interval to be grouped as a part of a symbol thereby removing regularity in packet inter-arrival time. This defeats detection algorithms modelled by Markovian and other models dependent on regularity of occurrence of the event used to modulate the information onto the shared resource (Jiangtao et al.[2010] ; Serdar et al.[2006,2008] ; Ahsan [2002] ; Hu 1992)

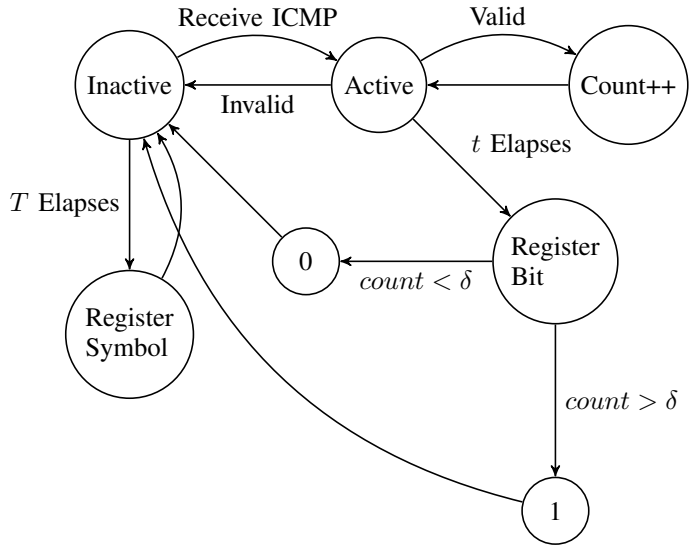


Fig. 2. Finite State Diagram for a Receiver Side Simple Binary Modulation Scheme Network Timing Covert Channel

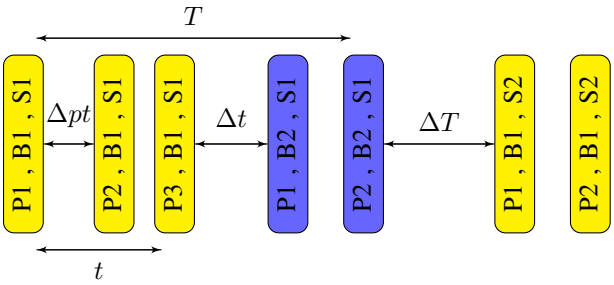
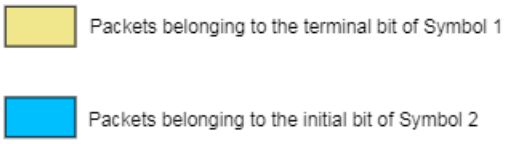


Fig. 3. Timing Diagram - *P* refers to the packet number, *B* refers to the bit number and *S* refers to the symbol number



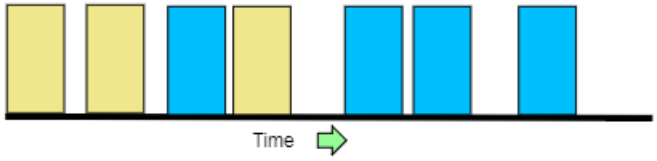


Fig. 4. A scenario where Δt is greater than or equal to ΔT

A. Arguments for using Hybrid Storage Covert Channels

In this subsection, we present our arguments for preferring *Storage Covert Channels* against *Timing Covert Channels*.

Definitions:-

T = The symbol period within which all the bits required for identifying the symbol are required to arrive.

t = The bit period within which all the packets required for identifying the bit are required to arrive.

Δt = The inter-bit arrival period.

ΔT = The inter-symbol arrival period.

Δpt = The inter-packet arrival period.

Binary Modulation Scheme Timing Covert Channel

The most basic implementation of a *Timing Covert Channel* includes a binary modulation scheme(Figure 2) where a burst of ICMP packet arrivals is treated as a high bit and a reduced frequency of arrivals is treated as a low bit where the frequency of arrival is compared against a certain δ set by the user to make the decision. The requirement for a time out Δt stems from the fact that IP protocol does not guarantee a time of delivery which implies that if packets pertaining to two bits are sent successively without any interval between them, there is a possibility that a packet belonging to the second bit could arrive in the time frame allotted to the packets of the first bit hence causing the receiver to deduce incorrectly. Whilst Δt is used for discerning between groups of packets in-order to deduce a bit, ΔT is used for discerning between groups of bits in-order to deduce a symbol. ΔT has to be higher than Δt to prevent discrepancies at sequence edges (Figure 4).

This model is analyzed by likening it to FSK(Frequency Shift Keying) modulation.In binary FSK, the carrier shifts between two sinusoids of different frequencies to represent a high and low bit. The high bit is generally associated with the sinusoid of higher frequency while the low bit with that of lower frequency. Non-Coherent detection of Binary FSK uses two filters(a high pass and low pass) tuned to the two carrier frequencies to isolate the respective carriers and then these are passed onto an envelope detector and their pulse shapes then compared to discern if its a high or a low bit. Similarly, the state *Register Bit* of Figure 2 acts as the filters where the cutoff frequency is determined by δ. If the frequency of packets (number of packets within t) is greater than the cutoff δ then its classified as a high bit and if not then its grouped as a low bit. If Δt increases, throughput decreases since the inter-bit arrival time increases(delay between groups of packets required to transmit each bit increases hence decreasing symbol arrival rate) but the rate of incorrect deduction at the receiver also decreases.Whereas if δ is decreased, the number of packets required to deduce a low bit decreases which due to packets getting dropped at intermediate nodes causes the rate of incorrect deduction at the receiver to increase.

In-order to compute the throughput, a coding scheme has to be developed using which a unique optimized binary sequence can be assigned to every symbol(alphabets in this case). We restrict ourselves to English alphabets in this

paper with the aim being to transmit messages in English only. Such a scheme can be developed by assigning codes to alphabets based on their probability of occurrence in a given passage, this is done with a view to transmit symbols with minimum number of bits to improve throughput. The frequency of occurrence of English alphabets have been found to adhere to the Pareto distribution (Li 1992). One example of a code that takes note of the alphabets's Pareto distribution and assigns sequences is the *Morse Code*. The "dits" are given by a low bit(0) and the "dahs" by a high bit(1) where each bit is discerned in accordance with the mechanism given in Figure 2. The "dits" are assigned a low bit since the number of "dits" in *Morse Code* for the 26 alphabets outnumber the number of "dahs" and since the low bit requires lesser number of packets to be transmitted, the regularity of packet arrival time (Δpt) can be reduced to subvert detection.

Let $L(S_k)$ be the length of the *Morse Code* sequence of the k^{th} symbol and $p(S_k)$ be the probability of transmission of the k^{th} symbol drawn from the *Pareto Distribution*. If L_{avg}, L_{max} and L_{min} give the average length of the symbol to be transmitted, maximum and minimum length of the code assigned respectively, then-

$$L_{avg} = \sum_{k=1}^{26} p(S_k)L(S_k) \tag{1}$$

$$L_{max} = max\{L(S_k)\}, \forall k \in \{1, 26\} \tag{2}$$

$$L_{min} = min\{L(S_k)\}, \forall k \in \{1, 26\} \tag{3}$$

The time t required to receive a bit cannot be varied and has to be decided before the channel is setup since changing t implies that the count of packets compared against δ changes thereby leading to incorrect deduction at the receiver. The time T required to receive a symbol also cannot be varied and has to be fixed before the channel is setup since, for example, the *Morse Code* for "a" is subsumed by the *Morse Codes* for "j", "l", "p" and "r", therefore a variable T could cause the receiver to incorrectly deduce a symbol.
As a consequence of fixed T and t, Δt and Δpt are subject to the condition -

$$\sum_{i=1}^{L_{max}} \Delta t_i < T - L_{max}t \tag{4}$$

$$\sum_{i=1}^{max\{n\}} \Delta pt_i < t \tag{5}$$

Where $n < \delta$ for low bit and $n > \delta$ for high bit. Therefore , the inter-bit arrival period Δt can be randomized subject to the condition given in (4) whereas, the inter-packet arrival period Δpt can be randomized subject to the condition given in (5). The throughput R is defined as *the amount of information transferred, over the time taken to transfer it*. Therefore the throughput in transferring N symbols is-

$$R = \frac{NL_{avg}}{NT + \sum_{i=1}^{N-1} \Delta T_i} \tag{6}$$

Re-writing (6) with T as the subject -

$$T = \frac{L_{avg}}{R} - \frac{\sum_{i=1}^{N-1} \Delta T_i}{N} \tag{7}$$

Substituting for T from (4) -

$$L_{max}t + \sum_{k=1}^{L_{max}} \Delta t_k < \frac{L_{avg}}{R} - \frac{\sum_{i=1}^{N-1} \Delta T_i}{N} \tag{8}$$

Re-writing (8) with R as the subject -

$$R < \frac{L_{avg}}{L_{max}t + \sum_{k=1}^{L_{max}} \Delta t_k + \frac{\sum_{i=1}^{N-1} \Delta T_i}{N}} \tag{9}$$

(9) gives the upper bound on the throughput attainable. The ΔT defines the inter-symbol arrival period. We have seen that T cannot be varied once decided after setting up the channel, therefore the minimum inter-symbol arrival period must be greater than the duration required to transmit a symbol of minimum code length subtracted from T to ensure that bits belonging to the next successive symbol are not transmitted in the time slice allotted to the current symbol(Figure 4). -

$$\Delta T_{min} > T - L_{min}t - \sum_{k=1}^{L_{min}} \Delta t_k \tag{10}$$

Re-writing (10) with T as the subject -

$$T < \Delta T_{min} + L_{min}t + \sum_{k=1}^{L_{min}} \Delta t_k \tag{11}$$

Substituting for T in (7)

$$\frac{L_{avg}}{R} - \frac{\sum_{i=1}^{N-1} \Delta T_i}{N} < \Delta T_{min} + L_{min}t + \sum_{k=1}^{L_{min}} \Delta t_k \tag{12}$$

Re-writing (12) with R as the subject -

$$R > \frac{L_{avg}}{\frac{\sum_{i=1}^{N-1} \Delta T_i}{N} + \Delta T_{min} + L_{min}t + \sum_{k=1}^{L_{min}} \Delta t_k} \tag{13}$$

(13) defines the lower bound of R. We now proceed to the discussion of techniques which would result in the detection of such a binary modulation scheme *Timing Covert Channel*. Lets assume Alice and Bob have a covert channel setup between their systems. Eve can sniff the traffic flowing into and out off Alice's system and Eve is aware that the English alphabets follow a *Pareto Distribution*. There are two possible detection strategies

that Eve can proceed with to detect the covert channel, one where Eve is always suspicious about Bob and Alice covertly communicating and the other where Eve is not suspicious but cautious of such a covert communication capable of taking place between Bob and Alice.

The always suspicious case refers to the detection strategy which can be incorporated by a detection algorithm with access to very large resources(both in time and processing speed) whereas the cautious case can be incorporated when the detection algorithm is limited by resources. Regardless of Eve's suspicion, the only observable features of the covert channel are Δpt and Δt as they are constrained by equations (5) and (4) respectively. Supposing Bob transmits N symbols to Alice through the covert channel, the total time required for such a transmission to take place is -

$$T_{total} = \frac{NL_{avg}}{R} \tag{14}$$

Owing to the alphabets following a *Pareto Distribution*, about 80% of these N symbols will be accounted for by 20% of the alphabets. The frequency of k^{th} alphabet is given by -

$$F_k = Np(S_k) \tag{15}$$

Since probability of occurrence of "e" is the most, majority of the regularity observed in Δpt and Δt will be pertaining to "e". Based on this observation, we can now bifurcate and devise Eve's two detection strategies -

Eve's Cautious Case

1.Choose an observation time interval O_t randomly.
2.Setup regularity threshold γ.
3.Look for regularity in Δpt and Δt within O_t.
4.Assign "e" to the most regular Δpt and Δt sequence.
5.If frequency of "e" assigned sequence is greater than γ, then covert channel is present.
6.If in the next O_t, a different sequence appears more regular, reassign "e".
7.If no regularity is found, increase O_t/decrease γ and repeat from step 3.

The total number of symbols expected to be transmitted in k^{th} O_{tk} is given by -

$$M_k = \frac{RO_{tk}}{L_{avg}} \tag{16}$$

The criterion for detecting the presence of a channel is then -

$$F_e = M_k p(S_e) \geq \gamma \tag{17}$$

Where F_e and $p(S_e)$ are the expected frequency of occurrence of "e" in M_k and probability of "e" respectively. The expected time required for detecting the covert channel can then be defined from (16) and (17) by substituting F_e as γ -

$$T_{detection} \geq \frac{\gamma L_{avg}}{p(S_e)R} \tag{18}$$

From (18) its clear that the minimum detection time $T_{min-det}$ for the *Cautious Case* is given by-

$$T_{min-det} = \frac{\gamma L_{avg}}{p(S_e)R} \tag{19}$$

Eve's Suspicious Case

1.Let T_o be the time the channel has been continuously monitored for.
2.Setup regularity threshold γ.
3.Look for regularity in Δpt and Δt.
4.Assign "e" to the most regular Δpt and Δt sequence.
5.If frequency of "e" assigned sequence averaged over T_o is greater than γ, then covert channel is present.
6.If no regularity is found, decrease γ and repeat from step 3.

Since in this case the observation duration is continuous, the expected number of symbols M at T_o will be a function of throughput R due to Bob changing his transmission rate over the course of T_o. Let this function be given by-

$$M = \frac{v(R, T_o)}{L_{avg}} \tag{20}$$

The frequency of "e" assigned sequence is then given by-

$$F_e = p(S_e)M = \frac{p(S_e)v(R, T_o)}{L_{avg}} \tag{21}$$

The criterion for detecting the presence of a covert channel is then obtained by averaging F_e over the entire observation time T_o -

$$\frac{p(S_e)\int_0^{T_o} v(R, T_o)dT_o}{L_{avg}T_o} \geq \gamma \tag{22}$$

Equation (22) defines the criterion for detecting the covert channel for *Eve's Suspicious Case*. When R does not vary over T_o as in the case of *Cautious Case* due to the observation time being small, the function $v(R, T_o)$ becomes a linear product of R and T_o which reduces equation (22) to equation (18). Therefore, as per (19), the minimum detection time is inversely proportional to the throughput R which is the only parameter that Bob can change by changing the transmission rate to evade detection apart from L_{avg} which we are assuming to be fixed. If Bob decides to evade detection by increasing ΔT, from (9), the throughput R decreases. Whereas if Bob decides to increase throughput by decreasing ΔT, $T_{min-det}$ decreases and hence risks compromising the channel. $T_{min-det}$ approaches infinity as R approaches 0 implying that given a large enough O_t, Bob and Alice's covert channel will eventually be discovered as governed by (19) unless they stop transmitting altogether.

Binary Modulation Scheme Storage Covert Channel

We have seen from the previous discussion that in *Timing Covert Channels* the regularity in Δpt and Δt can be exploited to detect the channel. The regularity in inter-packet arrival period Δpt can be reduced by opting for a *Binary Header Length Modulation Scheme Storage Covert*

Channel. In this scheme, the bits namely high and low are transmitted by transmitting packets with specially crafted header. There is still the issue of packets potentially getting dropped. This can be mitigated by using a scheme similar to the previously discussed *Binary Modulation Scheme Timing Covert Channel* wherein a bit arrival period t is defined and all duplicate crafted packets that arrive in this interval are treated as representing the same bit. While the former used packet count to distinguish a high and low bit, the latter depends on the crafted header to discern the bit while disregarding duplicate crafted packets arriving within one t.

The need for t stems from the aspect of duplicate packets because, say suppose two high bits are to be transmitted, how would the receiver realize whether this is a duplicate transmission or packets corresponding to a new bit? The packets can be referenced by an identification number to avoid the confusion however that would make it all the more easier to detect the channel. The other ways of handling packet dropping includes acknowledgements from the receiver but that would increase the regularity of the traffic and hence is not discussed.

Rules of the scheme can be outlined as follows-

Rule 1.
Receiver starts clock at the instant it receives a specially crafted packet and stops the clock after a duration t has elapsed to register the bit pertaining to the crafted packet.

Rule 2.
A single packet alone can be acknowledged by the receiver within t. Any duplicates past the acknowledged packet are disregarded by the receiver. Thus Δt the inter-bit arrival period can be decreased. This is explained as follows - Consider a case where Bob requires to transmit a message to Alice through a covert channel. Bob transmits a couple of crafted packets corresponding to the first bit and Alice receives it and starts the interval t. If Bob transmits crafted packets corresponding to the second bit immediately, then its quite possible that crafted packets corresponding to the second bit can arrive within the same t where the packets corresponding to the first bit are expected to arrive. But since the receiver has already acknowledged a packet corresponding to the first bit it will simply disregard any duplicate/original crafted packets within that t thereby avoiding discrepancies in bit determination.

Rule 3.
As with *Binary Modulation Scheme Timing Covert Channels*, the symbol arrival period T has to be fixed owing to the problems with discerning symbols due to the *Morse Code* used as discussed before.

This is essentially the basic working of a *Binary Modulation Scheme Storage Covert Channel*. Although its better than *Binary Modulation Scheme Timing Covert Channel* with its mitigation of regularity in Δpt by giving Bob complete freedom in deciding the number of duplicate packets to transmit, the issue concerning fixed t and T is still present. The throughput bound of *Binary Modulation Scheme Timing Covert Channels* from equations (9) and (13) is given by-

$$R > \frac{L_{avg}}{\frac{\sum_{i=1}^{N-1} \Delta T_i}{N} + \Delta T_{min} + L_{min}t + \sum_{k=1}^{L_{min}} \Delta t_k} \tag{23}$$

$$R < \frac{L_{avg}}{L_{max}t + \sum_{k=1}^{L_{max}} \Delta t_k + \frac{\sum_{i=1}^{N-1} \Delta T_i}{N}} \tag{24}$$

The only major difference in the throughput of *Binary Modulation Scheme Storage Covert Channels* will be the fact that there is no agreed upon δ hence n (the number of packets transmitted within t)is completely at Bob's discretion provided Bob is careful enough such that all the duplicate packets get consumed within a single t. For example, suppose Bob transmits n specially crafted packets with all of them corresponding to the same bit. If the m^{th} packet gets acknowledged by Alice, she starts t from this instant. The remaining $(n-m)$ duplicates must arrive before t elapses to prevent the duplicates from being recognized by Alice as another bit sent by Bob-

$$\sum_{k=1}^{n-m} \Delta pt_k < t \tag{25}$$

Therefore, in the case of Timing Covert Channels, the lower bound of t is fixed by equation (5) whereas t is fixed by equation (25) for Storage Covert Channels. Since a lower n is required for Storage Covert Channels, Δt can be increased at the expense of t from equation (4). This implies that from equation (23) and (24) by increasing Δt the R decreases but an NL_{avg} amount of bits are still being communicated just with lower number of packets. Since the "e" frequency detection test is still applicable to Storage Covert Channels, a decrease in R implies an increase in detection time from (18) for the same NL_{avg} amount of bits as compared to the case of Timing Covert Channels. Therefore, we conclude by proposing that for transferring the same amount of information since Timing covert channels require a larger number of packets the detection time is smaller than in the case of Storage covert channels since as n increases so does the regularity in Δpt.

Modified Structured Storage Covert Channels

From the discussion on *Binary Modulation Scheme Timing* and *Storage Covert Channels*, its quite clear that the regularity in Δpt and Δt arises from the need for fixed t and T. t and T can be chosen to be as large as possible which from equations (9,10 and 13) increases the detection time but also decreases R. To overcome this problem, we propose a modification to structured Storage Covert Channels where every packet is used to communicate one or more symbols. The lack of directly transferring

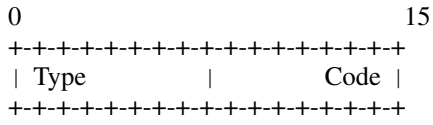


Fig. 5. ICMP Common Header

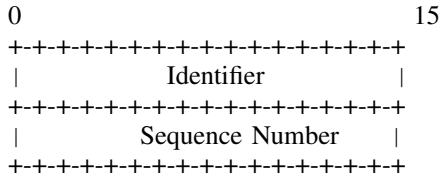


Fig. 6. ICMP Echo Packet Header

symbols in the case of the binary modulation scheme timing and storage covert channels dictates the need for fixed t and T due to the discrepancies in discerning bits and symbols respectively. But by using packets to transfer entire symbols, there is no need for fixed t and T which implies that Δpt can be completely randomized without compromising R. A simple method of achieving this will now be devised to support our claims but advanced methods of implementing it is out of the scope of this paper.

The ICMP Echo packet as shown in Figure 6, has two 16 bit fields namely *Identifier* and *Sequence Number*. The maximum value that can be held by the sequence number field is ”65535”.This value will be used to devise a offset-index based system to transmit at-most two symbols per packet. All operations henceforth mentioned are in the decimal system.

Sequence Number Field: The first number is used to index the base alphabet, the next number is used to indicate the offset of the to be transmitted alphabet from the base alphabet. The next two numbers are used along the same manner for indexing the base and for denoting the offset while the last number is added to the first base number or to the second base number based on the state of the identifier field.

Identifier Field: If a factor of 2 is transmitted in the identifier field then the last number of the sequence number field is added to the second base-offset number while if a factor of 3 is transmitted then the last number is added to the first base-offset number while if a 1 is transmitted then its added to both the base-offset numbers while if a prime excluding 1,2 and 3 is transmitted then no addition takes place.

In Figure 7, the highlighted ICMP Echo Packet has a sequence number ”10991” and a identifier number ”2”. From our previously discussed scheme, the first number of the sequence field denotes the first base number in this case ”a”, the next number denoting the offset from the first base is zero. The third number ”9” denotes the second base in this case the letter ”i” and the fourth number denotes the offset

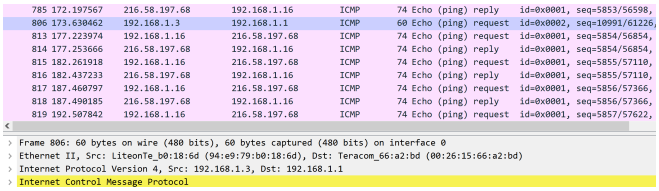


Fig. 7. Example for transmitting ”as” using the devised scheme

from the second base. Since the identifier field contains a 2 which is a factor of 2 hence the last number 1 of the sequence field is added to the second base-offset which is the fourth number of the sequence field. Therefore the second symbol being transmitted is (9+1) offset from the base ”i” which is the letter ”s” whereas the first symbol being transmitted is (0+0) offset from the base ”a” hence is the letter ”a” itself . Therefore the word transmitted is ”as”. Every alphabet can be denoted in 26 different ways using the base-offset indexing system which implies that to transmit a single symbol, an assortment of 26 different variations of packets can be used.Therefore, the word ”as”, can be transmitted in 676 different ways one symbol at a time. This randomization is further enhanced by the fact that since a fixed t and T is no longer needed, Δpt and Δt can be completely randomized. Though the algorithm used for transmitting the symbols is weak, unless Eve knows about the algorithm, the possibility of Eve detecting the channel between Bob and Alice is very slim. The Binary modulation schemes required a duration T to transmit each symbol whereas we require a single packet to transmit two symbols using this scheme therefore the throughput has increased while the risk of detection has decreased.

B. Vulnerabilities

In this section , we document the vulnerabilities which make the attack discussed in this paper possible.

B.1. Vulnerability in Software. Winpcap is quite a famous tool when it comes to computer networking , network virtualization etc . Although the tool was riddled with bugs [22][23][24] which allowed local users to execute code with administrator privileges, these bugs were fixed. When softwares like Wireshark or GNS3 etc which depend on Winpcap are installed, they by default install Winpcap . During the installation, the option “Start Winpcap at windows boot” (Figure 8) is selected by default. This default setting basically instructs NPF.sys driver of windows to start at boot without the need for administrator to manually load it hence any other software which requires Winpcap will not require elevation of privilege to start Winpcap since its already been started at windows boot. Using this first vulnerability we are able to run the malware which sniff for ICMP or any-other packets without elevation of privilege.

B.2. Vulnerability in Protocol. Our next vulnerability is inherent in the way ICMP protocol is designed. ICMP

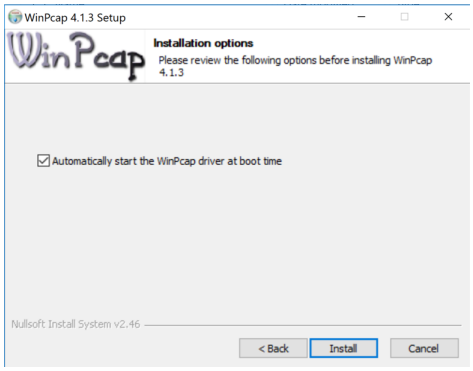


Fig. 8. Winpcap installation prompting activation of NPF.sys driver on Windows startup

is primarily used for network layer debugging and router advertisements(Narten *et al.* 2007). The specific message we use is ICMP Echo Request(Type 8 Code 0) and ICMP Echo Reply(Type 0 Code 0) however any other ICMP message which elicits a response at the destination such as Timestamp request, Information request etc can be used. When a system is pinged, the system replies but according to Postel.[1981] the data sent in echo request must be sent back in echo reply which basically means, supposing we send a particular payload as data in an echo request, the destination should ideally reply back with the same payload. This definition is the second vulnerability we exploit.

B.3. Vulnerability in Anti-Malware Software. Though anti-malware software on victim's side do not allow ICMP through, they have to make an exception for those ICMP sourced from the router. This is because the router advertises itself to the victim using ICMP Router Advertisements. This implies that anti-malware software cannot filter out ICMP messages from its gateway router. This loophole in anti-malware software which allows ICMP through in exceptional cases is exploited.A SANS study(Couture 2010) documents other exceptional situations in which firewalls and anti-malware softwares allow ICMP traffic through and all these cases also provide a vulnerable environment for the exploit discussed in this paper.

III. SETUP

With the three vulnerabilities at hand, we can now define the setup and flow diagram. The Winpcap vulnerability dictates that we don't really require administrator privileges to run programs that depend on Winpcap and the ICMP protocol vulnerability dictates that the data sent in ICMP Echo Request (Type 8 Code 0) must be returned in the ICMP Echo Reply(Type 0 Code 0).

.1. Victim Side. On the victim side, the victim unknowingly downloads the package. The package can be camouflaged as any file, executing it does not spook the user since because of the *Winpcap vulnerability* the package will not require

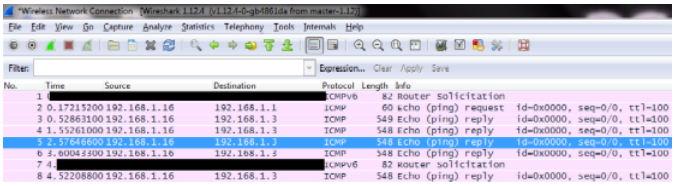


Fig. 9. File Piggybacking Demonstrated using *Unstructured Storage Covert Channels* and the Vulnerability discussed

elevation of privilege to gain access to NPF.sys driver. So unlike most ransomwares which do require elevation of privilege, the user in our case will not be prompted for administrator privileges and hence is completely unaware of the malicious code. A small section of code can also be added to make the package behave as the file it is imitating, to further mask suspicion. With the package downloaded and executed, it sets up a daemon which monitors ICMP messages waiting for valid payload. The attacker cannot just send an ICMP message carrying the payload to the victim since any anti- malware or in-situ defenders on the victim's computer would not allow it through as is the case with most Windows systems by default. This is discussed in the next subsection.

.2. Attacker Side. On the attacker side, the attacker crafts an ICMP Echo Request(Type 8 Code 0) packets and through the crafted ICMP packets, the attacker delivers the malicious payload to the gateway router whilst spoofing the victim one or more symbols at a time. This request elicits a legitimate reply from the gateway router which as explained under *Vulnerability in Protocol* must contain the crafted header. This reply is delivered to the spoofed victim. The anti-malware software on the victim side allow this ICMP reply to pass through to the daemon setup by the package since as explained under *Vulnerability in Anti-Malware Software*, the anti-malware software must allow ICMP from gateway to pass through. This ensures that the symbol transmitted by the attacker reaches the receiver which then concatenates it until it becomes a viable command and executes it.

IV. CONCLUSION

The throughput against detection time formulations based on the "e" frequency test approach decisively shows that traditional Storage Covert Channels can be made less regular than Timing Covert Channels. Although Unstructured Storage Covert Channels with massive payload carrying capability serve as an attractive medium for setting up Covert Channels, with ICMP, they can be identified easily. The Modified Structured Covert Channels introduced mitigate regularity by transferring one or more symbols as opposed to bits and can be made more robust by formulating better modulation schemes than the one discussed in this paper. Finally, as demonstrated in the paper, anti-malware softwares are as effective as their users are. Despite these systems having evolved to combat dated exploits, prevalence

of vulnerable yet popular softwares render them ineffective against certain dated exploits.

REFERENCES

[1] Lampson, B.W., "A Note on the Confinement Problem," *Communications of the ACM*, Oct.1973

[2] daemon9 and alhambra, *Phrack Magazine*.Volume Seven, Issue Forty-Nine,August 1996

[3] G. Fisk, M. Fisk, C. Papadopoulos, J. Neil, "Eliminating Steganography in Internet Traffic with Active Wardens," in *Proceedings of 5th International Workshop on Information Hiding*, October 2002.

[4] J. Patrick R. Gallagher, "A guide to understanding covert channel analysis of trusted systems," *National Computer Security Centre* NCSC–TG–030, no. Library No. S–240,572, 1993.

[5] Schaefer, B. B. Gold, R. Linde, and J. Scheid, "Program connement in KVM/370," in *Proceedings of the 1977 ACM Annual Conference*, (Seattle, WA), pp. 404–410, October 1977.

[6] P. A. Karger and J. C. Wray, "Storage channels in disk arm optimization," in *Proceedings of the 1991 IEEE Computer Society Symposium of Research in Security and Privacy*, (Oakland, CA), pp. 52–61, May 1991.

[7] W.-M. Hu, "Reducing timing channels with fuzzy time.," *Journal of Computer Security*, vol. 1, no. 3-4, pp. 233–254, 1992.

[8] Wendzel Steffen,Zander Sebastian,Fechner Bernhard,Herdin Christian, "Pattern-Based Survey and Categorization of Network Covert Channel Techniques," *Journal ACM Computing Surveys (CSUR)*, Volume 47 Issue 3, April 2015 Article No. 50

[9] J.C.Wray, "An analysis of covert timing channels," *Computer Society Symposium on Research in Security and Privacy*, 1991.

[10] R. A. Kemmerer, "Shared resource matric methodology: An approach to identifying storage and timing channels," *ACM Transactions on Computer Systems*,vol. 1, pp. 256–277, August 1983.

[11] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time," *Proc. 11th Usenix Security Symp., Usenix Assoc.*, 2002, pp. 149–167

[12] Erik Couture, "Covert Channels," *The SANS Institute,* August 19, 2010

[13] Kamran Ahsan, "Covert Channel Analysis and Data Hiding in TCP/IP," *Masters Thesis*, University of Toronto,2002

[14] SERDAR CABUK, CARLA E. BRODLEY, CLAY SHIELDS, "IP Covert Channel Detection"

[15] Joon Son, Jim Alves-Foss,2006.*Covert Timing Channel Analysis of Rate Monotonic Real-Time Scheduling Algorithm in MLS systems*.

[16] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts, "Covert Messaging Through TCP Timestamps." Massachusetts Institute of Technology - MIT, USA

[17] Serdar Cabuk, "Network Covert Channels: Design, Analysis, Detection, and Elimination," *Phd Thesis*, Purdue University, 2006.

[18] Song Li,A. Ephremides, "A covert channel in MAC protocols based on splitting algorithms," in *Wireless Communications and Networking Conference*, 2005 IEEE.

[19] Jiangtao Zhai, Guangjie Liu, Yuewei Dai, "A Covert Channel Detection Algorithm Based On TCP Markov Model," in *International Conference on Multimedia Information Networking and Security*, 2010.

[20] Postel,J, "Internet Control Message Protocol, DARPA Internet Program Protocol Specification," *RFC 792 page 15*, September 1981

[21] Narten, T., Nordmark, E., Simpson, W. and Soliman, H., "Neighbor Discovery for IP version 6 (IPv6)," *RFC 4861 page 10 -14*, September 2007

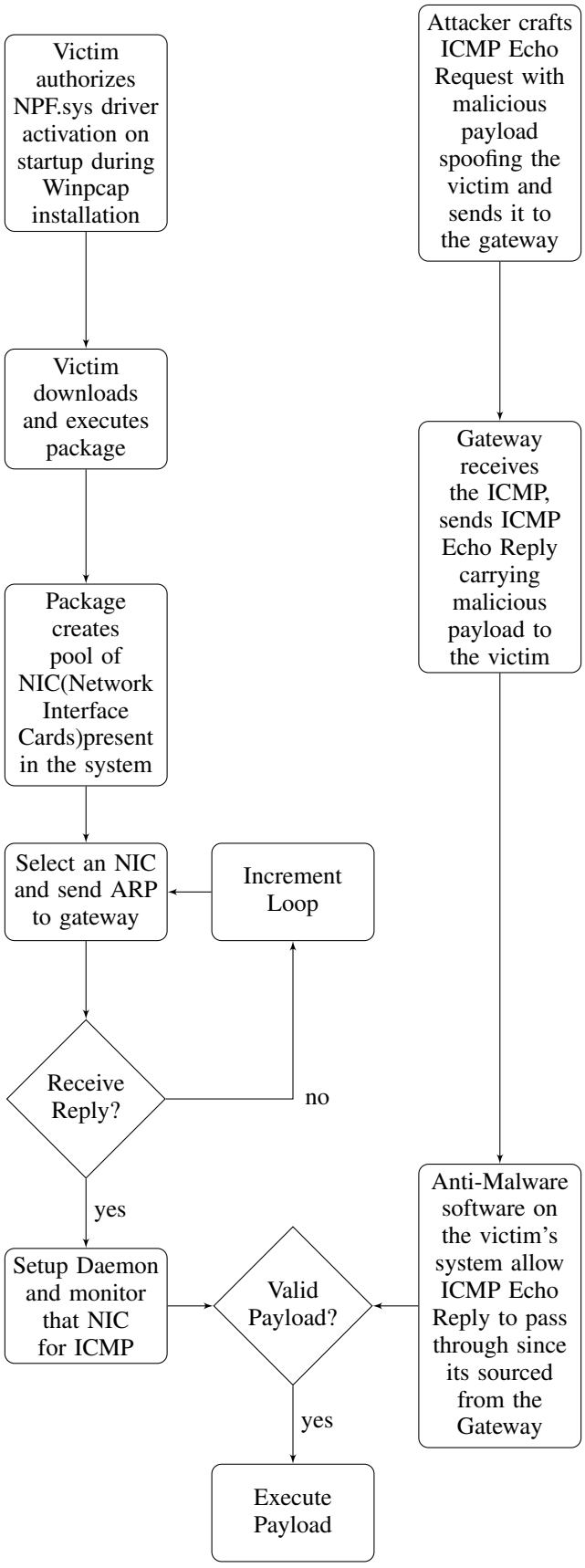


Fig. 10. Flow Chart

[22] Multiple array index errors in the `bpf_filter_init` function in NPF.SYS in Winpcap before 4.0.2, when run in monitor mode (aka Table Management Extensions or TME), and as used in Wireshark and possibly other products, allow local users to gain privileges via crafted IOCTL requests (July 13 ,2007) . Retrieved from https://www.cvedetails.com/cve/CVE-2007-5756/.

[23] The IOCTL 9031 (BIOCGSTATS) handler in the NPF.SYS device driver in Winpcap before 4.0.1 allows local users to overwrite memory and execute arbitrary code via malformed Interrupt Request Packet (Irp) parameters (July 11 ,2007) . Retrieved from https://www.cvedetails.com/cve/CVE-2007-3681/.

[24] Winpcap NPF.SYS driver code execution (July 9 ,2007). Retrieved from https://exchange.xforce.ibmcloud.com/vulnerabilities/35309.

[25] Checks for active machines to infect by sending an ICMP echo request, or PING, which will result in increased ICMP traffic(Febuary 13, 2007). Retrieved from https://www.symantec.com/security_response/attacksignatures/detail.jsp?asid=20407

[26] W. Li, "Random texts exhibit Zipf's-law-like word frequency distribution," *IEEE Transactions on Information Theory* (Volume: 38, Issue: 6, Nov 1992)