A Phase Space Analysis of Network Protocols

Tobias Blom
Institution for Computer Science
Linköping University
Linköping Technical University
tobbl213@student.liu.se

Charles Hamel
Institution for Computer Science
Linköping University
Linköping Technical University
chaha936@student.liu.se

Abstract

We investigate the problem of a third party being able to interfere into a TCP connection between two hosts by exploiting the predictability of the sequence number generation mechanism used by one of the hosts.

This paper is largely based on a previous work published in 2001 by Michal Zalewski. The original paper introduced the notion of Spoofing Set and described a method to build them by Phase Space Analysis in three-dimensional space. We extend this method by analyzing using a higher number of dimensions in order to reveal further weaknesses in the Pseudo Random Number Generators used in modern operating systems and embedded systems.

Our goal is to determine if in fact this ability to use higher dimensions is valuable from a practical point of view. We have found that in some cases, this gives a 50% increase in attack success probability. These results are tighly related to the specific PRNG used and thus no general conclusion can be made on the practical value of this method.

1 Introduction

The security of some protocols that the internet relies on, such as TCP and DNS depends on random number generation. In the case of TCP, preventing blind connection spoofing relies on the randomness of the initial sequence number (ISN). In DNS, spoofing is also prevented through the randomness of the query ID. A few years ago, a technique known as *delayed coordinates* was applied to attempt prediction of ISNs, and it was found that many implementations of TCP were very vulnerable to this kind of attack.

1.1 Goal

The practical goal of this paper is to investigate if some previously untested network equipment and embedded systems, such as PDAs, network attached printers and others, are vulnerable to a similar attack described in the earlier papers. We will in present our results on the analysis of selected network devices and their TCP/IP stack with special attention to their implementation of their pseudo-random number generator (PRNG). These devices are potentially much more vulnerable than the commonly used operating systems, mainly because the number of design constraints involved in their development, for example, the embedded processor might not be fast enough to compute good quality random numbers. We will also include the analysis of some current operating systems to hopefully confirm that they are no more vulnerable.

The theoretical aspect of our paper comes from the study of various aspects related to chaotic system analysis. In particular, we will study a phase-space reconstruction technique called *delayed coordinates* and its different parameters. We will base our research on the previous work done by Michal Zalewski [10], but will consider other implementations in hope of increasing the statistical value of the experiment. Our implementation will give us the ability to evaluate the method by extending it to *n* dimensions, or even suggest the optimal number of dimensions using a technique called *false nearest neighbor*.

1.2 Structure

The first part of this paper covers the basic principles needed to understand the method used for analysis, and it will also give a good description of the parameters influencing the results.

The second part will cover the tools used over the course of the research, followed a survey of several TCP/IP implementations, from modern operating systems to embedded systems that we have studied. We will conclude by analysing the effect of different parameters on probability of success of such an attack.

2 Methods and tools

This section will cover the background theory of the *delayed coordinates* method, with a discussion about the different parameters that effect the quality of the phase space reconstruction, such as: Embedding time delay, Embedding dimensions and the number of samples used for the reconstruction. It also covers a description of the original tools that were used as well as the new ones we created or altered in order to modify the embedding parameters.

2.1 TCP/IP sequence numbers

The TCP/IP protocol relies on a three way handshake process to establish a new connection [9]. This first step of this process occurs when a client wants to establish a new connection to a host on a specific port. It sends a packet with the SYN Flag along with the Initial Sequence Number (ISN), a 32-bit random number to the host. This packet is then acknowledged by the host which in turns generates its own random number (Acknowledgement Number), and sends it to the client. If the client receives a packet containing the ISN, it replies with a final acknowledgment to the host, containing the Acknowledgement number, and at this point the connection is established.

Sequence numbers are used to keep track of each packet and thus ensure session integrity in TCP connections. In order to protect against connection hijacking or data injection by a third-party host, the ISN generated by either the host or the client must be as unpredictible as possible.

The original RFC TCP specification [3] did not specify any requirement about the randomness of the initial sequence numbers. Thus, this aspect was often neglected by the system developers, resulting in implementations producing easily predictable sequence numbers.

In 2001, Michal Zalewski published a paper [10] reviewing PRNG implementations used in network operating systems at that time. Many of them were found to be vulnerable to blind-spoofing attacks, where an attacker could guess the ISN and interfere with the connection. The same author one year later published a follow-up report showing the progress made by operating system vendors. In many cases, the software providers had reacted promptly to the problem and that resulted in a much minute attack probability.

2.2 Phase space analysis

A phase space is a representation of all the possible states of a system, either linear or dynamic systems.

The generation of random numbers, such as in a good ISN generator is of the latter kind. In order to construct the phase space of a particular system we need to acquire a series of measurements at successive time intervals, called a time series. According to Thomas Scheriber, one of the authors of the TISEAN (TIme SEries ANalysis) package,

> "A time series can then be thought of as a sequence of observations performed with some measurement function. Since the (usually scalar) sequence in itself does not properly represent the (multimensional) phase space of the dynamical system, one has to employ some technique to unfold the multidimensional structure using the available data" [7]

This is what we intend to do by using the following methods.

2.2.1 Delayed coordinates

One of the most important methods used to reconstruct the phase space of a system is known as the "method of delays". The goal is to extract the hidden dependencies in subsequent ISNs generated by a host. This method suggests that we can achieve this goal by using previous samples of the sequence as additional coordinates in the space (thereof called the delayed coordinates).

The original paper suggests to use the first-order difference between samples in order to focus on the variation between samples, and not their absolute value. This method proved to be quite successful since many PRNGs are based on repetitive counter or time based algorithms, thus subsequent numbers are often only incremented. Supposing s a series of n subsequent ISN samples from a host, we can use the following equations to obtain a set of data points in the phase space.

$$\begin{aligned} z[n] &= s[n-2] - s[n-3] \\ y[n] &= s[n-1] - s[n-2] \\ x[n] &= s[n-0] - s[n-1] \end{aligned}$$

This method can be generalized to n dimensions. Theoretically, increasing the number of dimensions could provide deeper insight into dependencies between previous values. The original author suggested the use of three dimensions mainly because this was an ideal choice for visualization purposes [10]. Authors of the TISEAN package argue that the ideal choice of the number of embedding dimensions depends heavily on the application at hand. We will try to determine this in our analysis.

2.2.2 Attractors

Using the set of the (x, y, z) data points calculated previously, we can plot the phase space and get a visual representation of the complex behaviour of the random system. The fact that each and every point relies on current and previous values, potentially enables us to reveal patterns that would have gone unnoticed with a linear representation. These patterns created are called Attractors. Often unique, these abstract shapes in space illustrate where there are higher densities of data points, giving clues to flaws in the underlying PRNG algorithm.

Other than the number of embedding dimensions, we suggest the accuracy of the attractor also depends on the number of samples used in its creation. The original paper used the arbitrary number of 50,000 samples to reconstruct the phase spaces. This paper will try to determine the effect of increasing this number.

2.2.3 Spoofing set

The goal of the attacker is to be able to predict the next ISN from the host. Since the TCP/IP packet structure reserves 32 bits for the sequence number field, an attacker could flood the host using 2^{32} packets and achieve his goal. Fortunately, considering the minimum size of a SYN packet, which is 40 bytes, such an attack would represent 160 gigabytes of data transmitted in a short period of time, which is still hardly achievable with current technology.

Our goal is to build a set of ISN values that are the most probable of occurring, the choice of the size of this set is arbitrary but is directly related to the amount of bandwidth available to the attacker. The original paper suggested a size of 5,000 packets, considering the bandwidth improvement of consumer-grade internet connections in the recent years, a bigger spoofing set can be considered and this paper will explore this possibility.

Increasing the size of the spoofing set should intuitively increase our chances of guessing the next ISN, since adding possible values to the spoofing set will increase the probability that one of them is a correct guess of the next ISN. As stated above, there is a practical upper limit to the size of the spoofing set, but as network bandwidth and computer processing power steadily increase, we can increase the size of the spoofing set and analyze the effect of this.

2.2.4 Building a spoofing set using attractors

We can now use the attractor to build a good spoofing set; first, we must realize that the higher density areas in the phase space indicate a higher probability of finding the system in this state. These are the areas where we have a higher chance of guessing the right number. The key is in the set of equations given above, notice that the x coordinate of a point in space depends one the current ISN value $s[t]$ and the previous value $s[t-1]$. Thus if we were an attacker whose intention was to guess $s[t]$, the ISN value at the attack time.

This attacker would first have to query the host to get three subsequent samples $s[t-1]$, $s[t-2]$, and $s[t-3]$. Using these numbers, he could then calculate the y and z coordinates, putting a $m-1$ dimensional restriction on the possible values of the x coordinate. In the case of a three dimensional phase space, this would visually trace a line L, perpendicular to the x plane. In the case where m, the number of embedding dimensions is greater than three, L would be a plane, and the number of subsequent samples would be m.

Now that we have a set of possible x values, those that intersect with L. We can then manipulate the above equation to find suitable candidates for the spoofing set.

$$s[t] \quad = \quad x[t] - s[t-1]$$

Using the set of probable x values obtained at the previous step, and by knowing the previous number in the sequence, $s[t-1]$ we can include the value of $s[t]$ as a candidate in our spoofing set.

If we cannot find any points in the attractor intersecting with our restriction L, we use the generally accepted assumption about attractor that "if a sequence exhibits strong attractor behaviour, then future values in the sequence will be close to the values used to construct previous points in the attractor". Thus we introduce the $R1$ parameter, this initial, empirically selected radius reprensents the allowed distance from our restriction L within which we accept candidate values.

The size of $R1$ is to be chosen so that we can generate a non-empty spoofing set, i. e. it should not be too small but at the point that a spoofing set can be generated, there is no point in trying to increase the size of $R1$ any more.

In order to acheive the desired size of spoofing set, we need to introduce another parameter, $R2$. Still using the same reasoning, we assume that points close to our candidates also have a probability of being the next in the sequence. Thus, for every c candidate in the spoofing set, we include all the integer values from $c - R2$ to $c + R2$.

2.2.5 False nearest neighbors

A method to determine the minimum number of embedding dimensions in a phase space was suggested by Kennel et al. [6], a method called the false nearest neighbor method. This method relies on the fact that when decreasing the number of dimensions in the phase space, some data points that were distant in upper dimensions now look like neighbors by being projected in lower dimensions.

For example, suppose two points (x, y, z) in space, $p1 = (1, 1, 1)$ and $p2 = (2, 2, 101)$. The distance between those two points is described by equation 1:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} = \sqrt{10002} \quad (1)$$

If we decrease the phase space to two dimensions, then this distance becomes $\sqrt{2}$. Thus, $p2$ now looks like a neighbor point of $p1$, but it is a false neighbor, thus the method's name. In this lower dimension, $p2$ could possibly become a candidate in the spoofing set if indeed $p1$ was included in the intersection of the attractor and the restriction L. We intend to test this method in our analysis in order to determine the ideal embedding dimension.

2.3 Tools

The tools used in our analysis come from a variety of sources. First, we have used the *false_nearest* and *delay tools* from the TISEAN package. In addition, we have used the tools provided by Michal Zalewski, the author of the original paper on the topic. Parts of the original code has been modified for the purpose of this report, details are provided in section 4.

For collecting and recording sequence numbers a number of standard network tools and shell scripts were used. The command *tcpdump* was used to record the SEQ package received from the target device, and the command *nc* (netcat) was used to make a connection to the target computer. In general a *nmap* scan was performed to gather more information about the inspected device.

To visualize the delayed vectors, *gnuplot* was used to generate a 3D plot.

3 Vulnerability survey

This is the result of our work, presented with graphs from three dimensional delay vector analysis and data from the tools used, each device listed separately.

The input parameters *Dimensions*, $R1$, and *Spoofing set* are chosen and varied in the different test cases. Parameters *Average R2* and *Average N* are calculated in the analysis process, and points to the quality of the result of the test, and the parameter *Probability* is the goal function in all of the tests.

Dimensions (n) :	Number of embedding dimensions used for phase space reconstruction
R1:	Maximum allowed distance from L, the restriction plane
Spoofing set (ss):	Number of sequence numbers in the spoofing set
Average R2:	Proposed radius of second guess
Average N:	Average number of the guess set obtained
Probability:	The probability of guessing the correct sequence number

3.1 Operating systems

3.1.1 Windows 2000 SP3

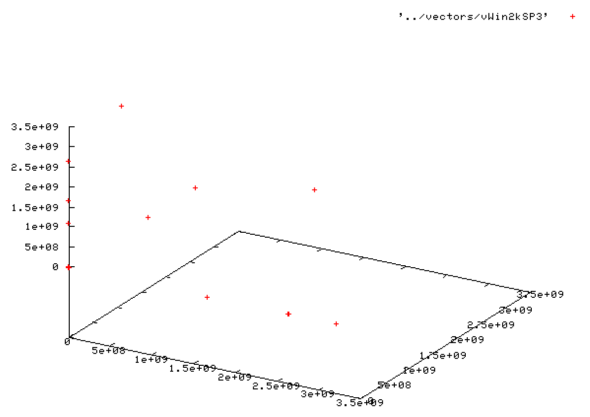

Figure 1. Windows 2000 SP3

Dimensions:	3	4	5	6
R1:	20,000	20,000	20,000	20,000
Spoofing set:	5,000	5,000	5,000	5,000
Average R2:	1	1	2	8
Average N:	42,132	20,640	25,617	6,464
Probability:	2%	5%	11%	15%

Dimensions:	3	4	5	6
R1:	20,000	20,000	20,000	20,000
Spoofing set:	20,000	20,000	20,000	20,000
Average R2:	4	8	12	49
Average N:	42,132	20,640	25,617	6,464
Probability:	3%	15%	45 %	55 %

Windows 2000 with Service Pack 3 is an excellent example that shows the effect of different parameters on the probability of success of the attack. First, observe the behaviour when we increase the number of dimensions from 3 to 6 using a spoofing set size of 5,000 to 20,000, in every case, the probability increases almost linearly according to the number of dimensions. Using a spoofing set of 5,000, the probability goes from 2% for 3D to 15% for 6D. A more drastic example comes when using a spoofing set of 20,000 packets, then it goes from 3% for 3D to 55% for 6D. Also note the effect of the spoofing set size, the larger it is, the higher is the probability. Comparing the result from a 6D 5,000 packets spoofing set to a 6D 20,000 packets set, we see that the probability more that doubles, going from 15% to 55%.

Dimensions:	3	4	5	6
R1:	1,000	1,000	1,000	1,000
Spoofing set:	20,000	20,000	20,000	20,000
Average R2:	232	5,625	9,374	9,999
Average N:	1,456	33	13	11
Probability:	55 %	40 %	4 %	1 %

Another important fact to notice in this particular case is the effect the parameter $R1$ has on the results. For instance, when using a spoofing set of 20,000 and $R1$ of 1,000, we achieve a very respectable 55% success rate, in 3D. If we augment $R1$ to 20,000, using the same spoofing set size, we obtain a much lower probability of 3% (Upper figure). On the other hand, we can see that using a 6D phase space reconstruction, the effect of this same increase has the inverse effect, the probability goes from a mere 1% to 55% (Upper figure). This last effect is easily explainable by considering equation 1.

We see that increasing the number of dimensions adds to the distance from the L restriction, thus augmenting the $R1$ allows a greater number of candidate values to be considered. In order to explain the sudden decrease observed in 3D, note that the Average N increases from 1,456 to 42,132. This means that at least half of the N candidates must be eliminated in order to fit the specified 20,000 spoofing set size. We can relate to this phenomenon as “having too many good choices”.

3.1.2 Windows XP SP2

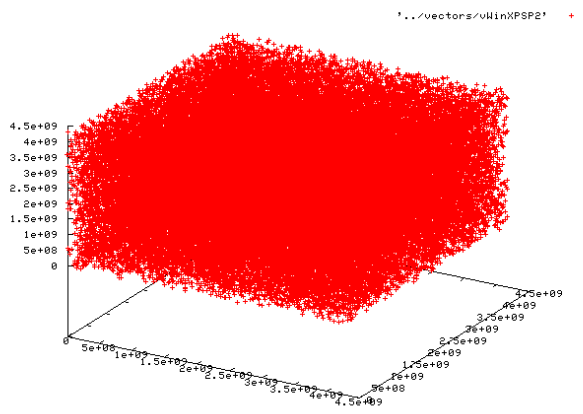

Figure 2. Windows XP SP2

Dimensions:	3	4	5	6
R1:	1,000	1,000	1,000	1,000
Spoofing set:	10,000	10,000	10,000	10,000
Average R2:	4,999	4,999	4,999	4,999
Average N:	11	10	10	10
Probability:	0%	1%	1%	6 %

Dimensions:	3	4	5	6
R1:	10,000	10,000	10,000	10,000
Spoofing set:	10,000	10,000	10,000	10,000
Average R2:	4,210	4,096	4,354	4,624
Average N:	15	15	14	12
Probability:	35%	17%	10%	13%

Dimensions:	3	3	3
R1:	10,000	10,000	10,000
Spoofing set:	5,000	10,000	20,000
Average R2:	2,072	4210	8567
Average N:	15	15	15
Probability:	19%	35%	50%

The first table shows the positive effect increasing the number of dimensions has on the attack’s success probability. This only seems valid for low values of $R1$, if we compare the results of Table 1 to Table 2, where $R1$ goes from 1,000 to 10,000, the effect is reversed, just as it was the case with Windows 2000 SP3. The last table shows that increasing the spoofing set size results in almost linear improvement of the attack feasibility.

3.1.3 Windows Vista

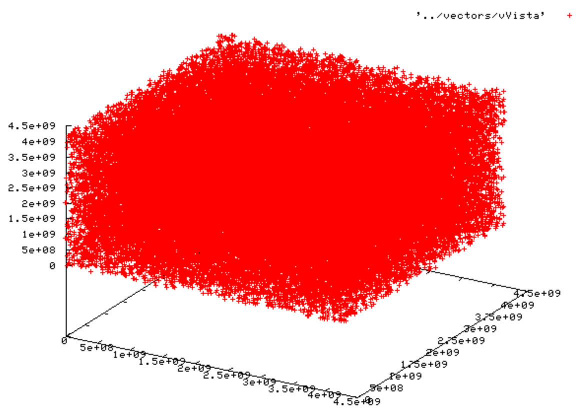

Figure 3. Windows Vista

According to areport on Windows Vista Network Attack Surface Analysis [4], Vista has reimplemented the ISN generator in accordance to the RFC 1948.

From the above figure, and from the results gathered from our tools, we can conclude that Windows Vista is not vulnerable to a blind-spoofing attack.

3.1.4 MacOS X 10.4.9

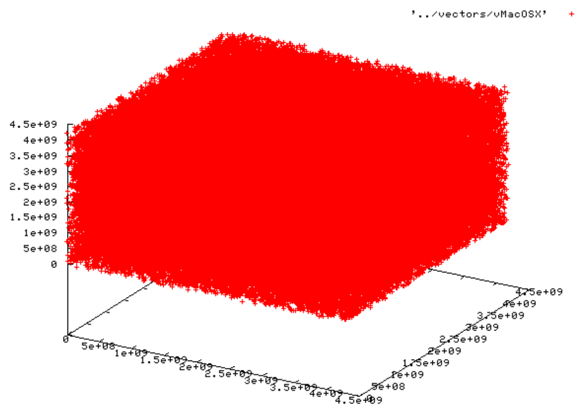

Figure 4. Mac OS X 10.4.9

Our tests show that the ISN generator used in MacOS X is not vulnerable to a blind-spoofing attack. All of our attempts have resulted in a 0% percent success rate.

3.2 Embedded systems

3.2.1 Apple AirPort Express

Running: Apple embedded
OS details: Apple Airport Express WAP v6.3

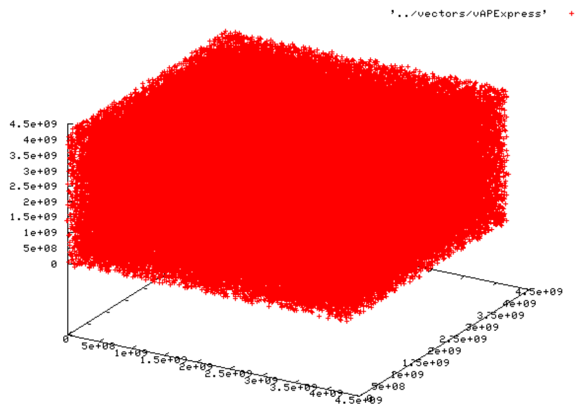

Figure 5. AirPort Express

Dimensions:	4	5
R1:	10,000	100,000
Spoofing set:	5,000	5,000
Average R2:	2,367	2,336
Average N:	11	12
Probability:	0%	0%

The AirPort Express is a common wireless access point by Apple [1]. According to *nmap* it runs Apple embedded OS.

Based on our analysis, we should mention that there are similarities between the Mac OS X's ISN generator output and the one found in the AirPort Express.

3.2.2 FON 2100 (Firmware version 0.7.1 r3)

Running: Linux 2.4.X|2.5.X|2.6.X, Belkin embedded
Firmware version 0.7.1 r3

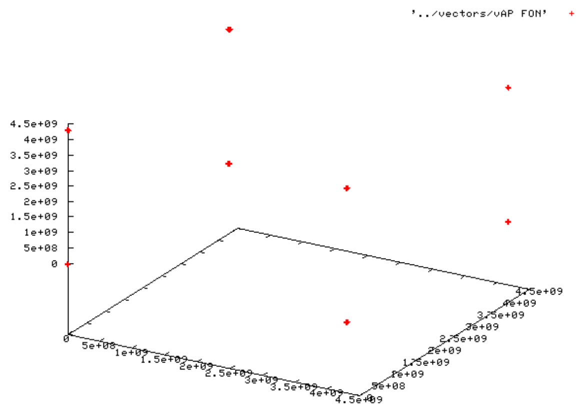

Figure 6. FON Wireless AP

Dimensions:	4	5
R1:	10,000	100,000
Spoofing set:	5,000	5,000
Average R2:	80	8
Average N:	422	3,807
Probability:	0%	0%

This device is a wireless access point sold by FON Wireless. It's according to *nmap* running some flavor of Linux.

The quality of the ISN generation is high, and although the graph does look trivial, we were never able to guess the correct sequence number.

3.2.3 Linksys WAP54G

```
Running: Linux 2.4.X|2.5.X|2.6.X
OS details:
Linux 2.4.0 - 2.5.20,
Linux 2.4.18 - 2.6.4 (x86),
Linux 2.4.19 w/grsecurity patch,
Linux 2.4.20 - 2.4.22 w/grsecurity.org patch,
Linux 2.4.22-ck2 (x86) w/grsecurity.org and HZ=1000 patches
```

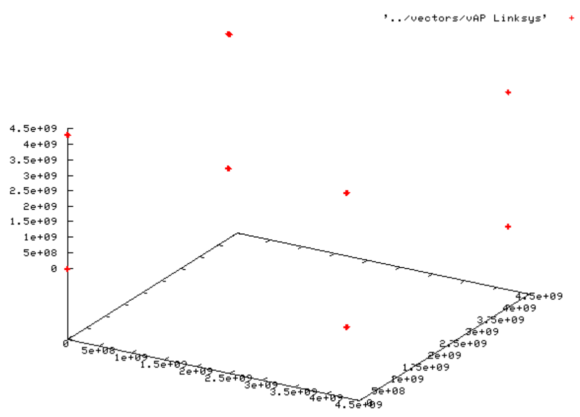

Figure 7. Linksys WAP54G

Dimensions:	4	5
R1:	10,000	10,000
Spoofing set:	5,000	10,000
Average R2:	89	172
Average N:	400	372
Probability:	0%	0%

The same goes for the Linksys WAP54G, running Linux with a good ISN generator output.

3.2.4 D-Link WBR-1310

```
Running: Linux 2.4.X, Maxtor Linux 2.4.X
OS details: Linux 2.4.20 - 2.4.32, Linux-based embedded device (Linksys WRT54GL WAP, Buffalo AirStation WLA-G54 WAP, Maxtor Shared Storage Drive, or Asus Wireless Storage Router)
```

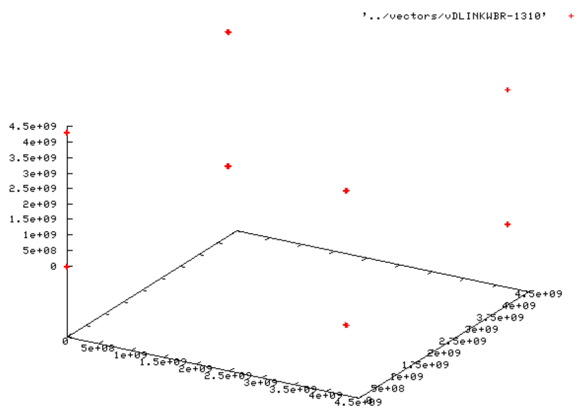

Figure 8. D-Link WBR 1310

Dimensions:	4	5
R1:	10,000	10,000
Spoofing set:	5,000	10,000
Average R2:	87	217
Average N:	315	293
Probability:	0%	0%

Yet another access point running Linux, with the same results as previously mentioned.

3.2.5 SMC

```
Running: SMC embedded
OS details: SMC Barricade DSL Router/Modem/Wireless AP
Runtime code version: V1.07.2
Model: SMCWEBT-G
```

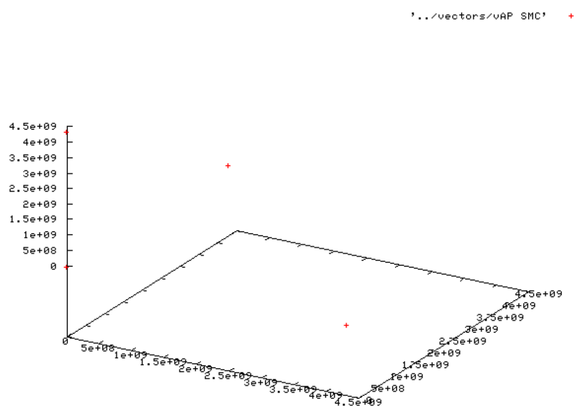

Figure 9. SMC Wireless AP

Dimensions:	3
R1:	10
Spoofing set:	50
Average R2:	0
Average N:	148
Probability:	100%

This is a very basic wireless access point by SMC Networks. It runs some sort of embedded OS, we can see that the ISN generation is very weak. In fact, it is based on a simple 16 bit counter. This device could easily be used to perform an idle portscan [5] by an attacker.

The fact that this device has implemented a trivial ISN generator does not actually mean that it's trivial to attack. The device also gives the user the choice of WPA2 encryption on the wireless interface, meaning that an attack like this is not possible unless the WPA2 encryption first is bypassed.

Still, the device is vulnerable both if no, or simple encryption on the wireless is first breached, or if the attacker tries an attack on the administrative interface, a web server available only through the wired network interface.

3.2.6 Axis 2100

Device type: Web camera
Running: Linux 2.0.X
OS details: Linux 2.0.34-38

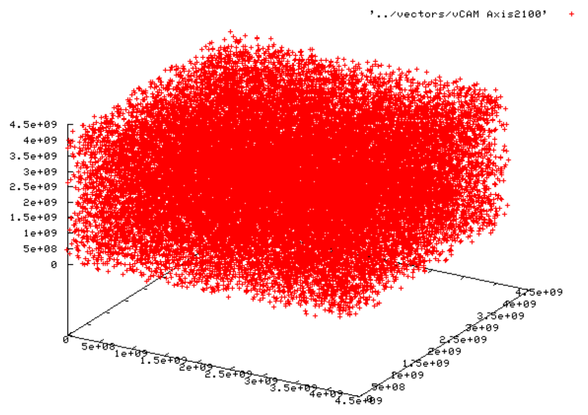

Figure 10. Axis 2100 Web camera

Dimensions:	4	4	4	5	5
R1:	100	1,000	10,000	100	10,000
Spoofing set:	5,000	5,000	5,000	5,000	5,000
Average R2:	2,481	2,309	2,163	2,481	2,159
Average N:	13	28	32	13	33
Probability:	22%	86%	87%	26%	87%

The Axis 2100 is a network-attached camera with an internal web server with FTP upload capabilites. It is running Linux kernel 2.0.

The above table shows the effect of increasing the size of $R1$, allowing more candidates to be considered in the spoofing set. It also shows that increasing the number of dimensions results in the inclusion of candidates previously unconsidered in lower dimensions, providing a 4% probability increase in 5D.

3.2.7 Linksys PPS1UW

Running: Linksys embedded
OS details: Linksys EtherFast Print Server

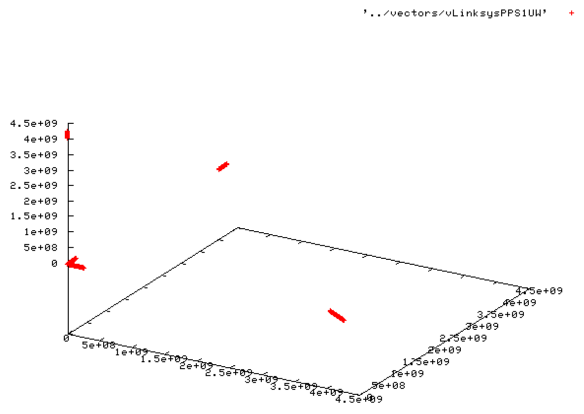

Figure 11. Linksys PPS1UW

Dimensions:	3
R1:	10
Spoofing set:	10
Average R2:	3
Average N:	18
Probability:	100%

The Linksys PPS1UW is a wired/wireless USB print server from Linksys. As seen on the graph, its ISN generator is trivial. One particular feature of this generator is that in most cases, its output is 0.

3.2.8 HP LaserJet 2100 Printer

Running: HP embedded
OS details: HP printer w/Jet Direct

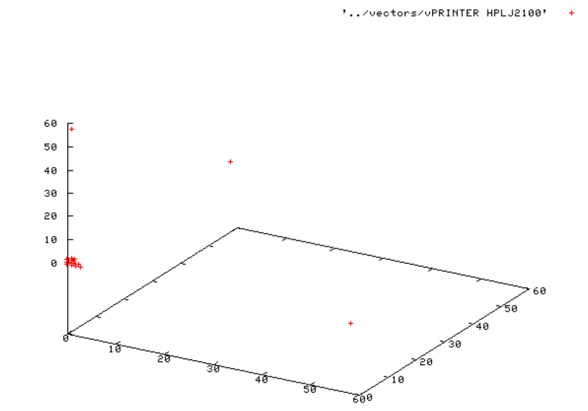

Figure 12. HP LaserJet 2100

Dimensions:	3
R1:	10
Spoofing set:	10
Average R2:	0
Average N:	18
Probability:	100%

This printer implements a trivial ISN generator, based on a simple 16 bit counter, but it sends each number four times - there is no entropy at all.

3.2.9 Palm Tungsten T (m550)

Device: Palm Tungsten T (m550)
Running: PalmOS software v.5.0
PPP NetIF v.8.0
Connected through Softick PPP 2.33

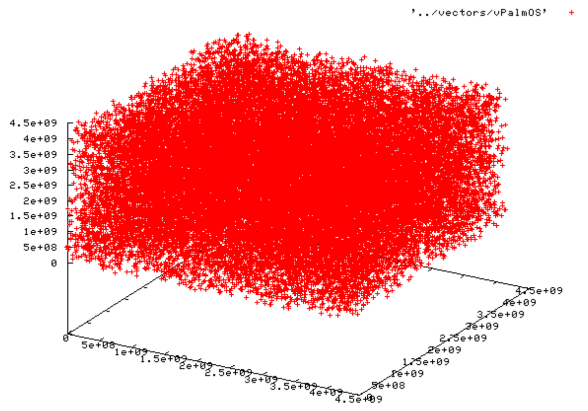

Figure 13. Palm OS

Dimensions:	4	5
R1:	10,000	10,000
Spoofing set:	5,000	10,000
Average R2:	87	217
Average N:	315	293
Probability:	0 %	0%

We have installed an FTP server on the handheld in order to extract ISNs from it. The device is connected through a Windows XP machine running Softick PPP 2.33 [8]. Note that we actually do not know if the packets are in any way altered when passing through the host machine, and if in this case we reveive incorrect results.

3.2.10 Draytek Vigor 2200e

Running: Draytek embedded
OS details: Draytek Vigor 2200e DSL router v2.1a

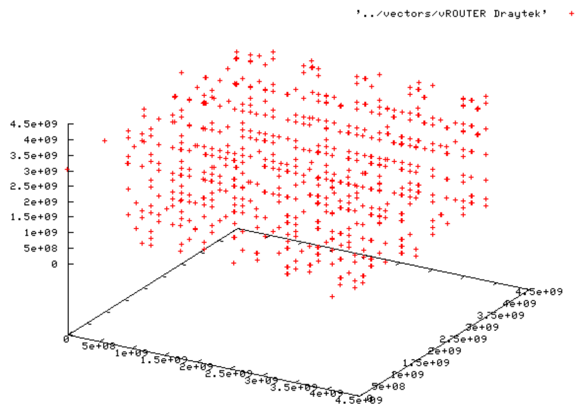

Figure 14. Draytek Virgo 2200e

This DSL router from Draytek [2], a Dutch company, has an interesting ISN generator pattern, widely scattered but still regular and ordered. This suggests a statistical bias of some kind. Despite this apparent weakness, we were unable to make a guess, even with very large values of $R1$.

Dimensions:	5	6
R1:	10,000	10,000
Spoof set:	10,000	20,000
Average R2:	0	0
Average N:	0	0
Probability:	0%	0%

4 Implementation

The original paper included a set of tools to accomplish all the tasks necessary to gather and analyze the data. The basic limitation of these tools was that they were specifically designed to work with 3-dimensional phase spaces. One of the goals of this paper was to explore if increasing the number of dimensions used for the phase space reconstruction would have any effect on the probability of guessing the next sequence number.

Most of the work has been done on the *guess3d* tool, now appropriately renamed *guessnd*. The first enhancement was related to the number of sequence numbers that can be used to reconstruct the phase space. This number is now arbitrary large.

The second enhancement is that it can now reconstruct the phase space in n dimensions, allowing us to observe the complex behavior of the PRNG at a deeper level, possibly revealing further hidden dependencies on time or other environmental factors.

In order to gather an ISN sequence from a target host, we used the original script provided by Zalewski using *tcpdump* to gather the SYN packages and requesting them by a network call using *netcat*.

The next step in the process is to visualize the data we gathered. We created a tool called *delayvector* which computes the n-dimensional vectors from the linear ISN sequence. The resulting points can then be plotted using *gnuplot*.

Finally, to determine the vulnerability of the different implementations, we wrote a simple shell script. Its role is to automate the testing process by launching different instances of the slightly modified *calprob* tool, varying the number of dimensions, spoofing set size and $R1$.

5 Risk analysis

After further research, it seems that the risks associated with an attacker being able to perform a successful ISN blind spoofing attack against a modern operating system is still present today. Some operating system vendors, such as Microsoft have implemented the RFC 1948 resulting in a 0% attack probability on Windows Vista. Unfortunately, their previous attempt, Windows XP with Service Pack 2 still remains much vulnerable to this kind of attack.

In our opinion, the strenght of this method comes from the fact that it uses the difference between subsequent ISN values in order to build the attractor, not the absolute values, capturing the behavioral pattern of the ISN generator. Thus an attacker could potentially acquire a set of ISN from another similar system and use that information to make prediction on the actual victim. It is also important to consider that in real life, the probabilites of success for a DoS attack againts a weak ISN generator would be even greater since any traffic within the TCP/IP Window Size is accepted by a remote host, potentially ranging from $-2^{16}$to $+2^{16}$from the next ISN [10]. In the real world, a sucessful blind-spoofing attack could result in altered email commucation, malicious code injection into HTTP requests, firewall rule bypassing or in a more general sense, every network protocol on top of TCP utilizing plain-text commucation [11].

6 Conclusion

The initial goals of this paper were to first investigate the vulnerability of a selection of embedded devices against an ISN blind spoofing attack. The second important goal was to discover if using a higher number of embedding dimensions in the phase phase reconstruction would help improve the attack success probability. We have been able achieve both of the goals using empirical tests.

Regarding embedded devices, most of them were found to be running a Linux 2.2 or 2.4 kernel, implementing one of the best ISN generator, as found in the original paper. Exceptions made of the SMC WEBT Access point and HP LaserJet 2100 which clearly proved that some manufacturers negliged this aspect of the design.

We would have liked to test devices based on the VxWorks operating system and industrial control units but were unable to get access to any of them.

We have also been able to quantify the effect of different parameters, including the number dimensions, on the probility of success of the attack. First, when attacking a weak ISN generator implementation, the spoofing set size has a linear improvement effect on the probability. Second, we have found that increasing the number of dimensions in the phase space reconstruction rarely produces better results than when using a 3-dimensional reconstruction. When it does increase the probability, it is tightly related to the $R1$ parameter, which can, in turn, be ajusted differently and give equivalent results in 3-dimensional space.

An important note to make is that the results are highly dependent on the particular ISN generator under study, we cannot make general conclusions on whether or not increasing the number of dimensions is desirable for all implementations, we can only conclude that it has the potential to affect the attack success, positively or negatively.

7 Acknowledgements

Thanks to David Byers at IDA, LiTH for the help, guidelines, and the time spent in the lab to make the test devices work.

References

[1] http://www.apple.com/.

[2] http://www.draytek.com/.

[3] Rfc 793: Transmission control protocol. http://www.ietf.org/rfc/rfc793.txt, September 1981. DARPA Internet Program.

[4] J. Hoagland. Windows vista network attack surface analysis. http://www.symantec.com/avcenter/reference/Vista_Network_Attack_Surface_RTM.pdf.

[5] Insecure.org. http://insecure.org/nmap/idlescan.html.

[6] Kennel, Brown, and Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction, 1992.

[7] T. Scheriber. http://www.mpipks-dresden.mpg.de/ tisean/.

[8] Softick. http://www.softick.com/ppp/.

[9] Wikipedia.org. Tcp three way handshake. http://en.wikipedia.org/wiki/Transmission_Control_Protocol#Connection_establishment/.

[10] M. Zalewski. Strange attractors and tcp/ip sequence number analysis. http://lcamtuf.coredump.cx/oldtcp/tcpseq.html.

[11] M. Zalewski. Strange attractors and tcp/ip sequence number analysis, one year later. http://lcamtuf.coredump.cx/newtcp/.