Disrupting time management in a Microsoft Windows 2008 Active Directory environment using NTP

Wouter van Dullink, Dennis Pellikaan, Tjebbe Vlieg, Pieter Westein
System and network Engineering, University of Amsterdam

Abstract

The Network Time Protocol has been around on the internet since as early as 1985. Its main use to synchronize system in a network. During these years, the protocol has not been changed significantly. In this research paper various attack strategies regarding NTP are examined and performed. The main focus lies on attacks from a man-in-the-middle position. Both authenticated as well as non-authenticated NTP packets are taken into account. Attacks that are described are a delay attack, packet modification and a denial-of-service attack. All experiments are done in a Microsoft Active Directory environment with systems running on Windows 2008 Server. The conclusion is that NTP is reasonably safe, if authentication by means of an authenticated message digest is used. If this is not the case, great clock skews can be obtained by modifying NTP replies. In case of authentication, a DoS attack can be successful by overloading the server with NTP requests which require a cryptographic hash to be computed.

Contents

1 Introduction

Network time protocol (NTP) is one of the backbone protocols in a time dependent network environment. Systems like Kerberos authentication, which is used by Microsoft Active Directory, heavily depend on time synchronization. Many time dependent software do not provide time synchronization tools their selves, but rely on protocols like NTP. NTP is merely used to synchronize time between systems in a network and as such security was not its main concern. Although NTP does feature some security mechanisms, there has been examples of implementations where these mechanism were not used. NTP merely provides these features, but they are not mandatory. It is therefore very plausible that certain networks are vulnerable to attacks concerning time synchronization of its systems.

Having these considerations in mind, this article will treat the following research question:

Is it possible to achieve a significant clock skew between two systems in a default Windows 2008 Active Directory environment, which uses NTPv3, by means of a man-in-the-middle attack?

In answering this research question, it makes sense to first pay attention to various (sub)questions. To begin with, attention to the specific working of NTP, with emphasis on version 3 (as Microsoft uses this version), has to be given. Second, the default configuration of NTP in a Windows 2008 Active Directory environment has to be found out. Furthermore, the way this specific environment handles clock skews must be treated. The final subquestion is how the local time of a specific client can be adjusted.

Microsoft Windows 2008 Active Directory has been chosen as a test environment because of the fact that it is widely used by companies both small and large. Another reason which makes this specific environment interesting is the fact that it uses Kerberos for authentication (as well as for authorization), which is, like stated above, highly dependent of time. Kerberos' Time Granting Tickets (TGTs) are for example supplied with an expiration time. Another example is that Kerberos uses a system's local timestamp as a nonce in encrypted messages.

This article is structured as follows: section 2 treats the background of NTP. Both the fundamental principles as security options are explained. In section 3 the followed approach is covered. Section 4 deals with the actual experiments that have been carried out, along with their results. Finally, in section 5 conclusions are drawn from the test results.

2 Background on NTP

Network Time protocol (3), also know as NTP has been developed by David L. Mills since 1985 and is widely available on the internet to synchronize system clocks between computers. Understanding the working of NTP is important in order to be able to create clock skews in a system. In this section, NTPv3 will briefly be explained, as well as the specific implementation by Microsoft.

2.1 The Network Time Protocol

Network Time Protocol uses a hierarchical model where each level consists of servers that act as time sources for lower level time servers. Such a level is called a 'stratum'. A stratum 0 device is a hardware clock, for example an atomic cesium clock. Each time server below in this hierarchy has an increased number: stratum 1 servers are attached to the hardware clocks and function as time sources for stratum 2 servers, and so on. Theoretically, there can be up to 256 strata, although only the first 16 are used in practice(6).

One of the downsides of this specific architecture is that each level may cause extra time difference between clocks because of unpredictable delay on the network between synchronizing systems. NTP provides three different metrics to take this difference into account (a full description of the NTP packet can be found in Appendix A:

Clock offset The real clock skew between the local clock and the reference clock (ideally, this should be zero).

Round trip delay The time a packet needs to travel from sender to receiver plus the time the reply packet needs to travel to the initial sender.

Dispersion The maximum error of the clock offset.

Offset and delay are calculated by the sending and receiving of respectively an NTP request and reply. Such a request consists of several fields, of which these are most important:

Originate timestamp Timestamp of the sender's local time at the moment of sending the NTP request.

Receive timestamp Timestamp of the time server's local time at the moment of receiving the NTP request.

Transmit timestamp Timestamp of the time server's local time at the moment of transmitting the NTP reply.

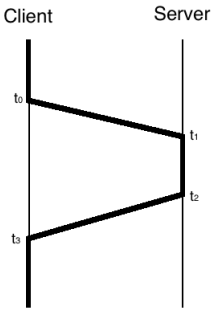

Fig. 1: Round trip NTP packets

Furthermore, the client makes a timestamp at the moment of receiving the NTP reply, leading to a total of four timestamps[1]. Figure 1 gives an overview of NTP packet roundtrip, as well as the specific timestamp moments.[2] These four timestamps are enough to calculate clock delay and offset, which is done using equation 1 (delay) and 2 (offset).

$$\delta = (t3 - t0) - (t2 - t1) \tag{1}$$

$$\theta = \frac{(t3 - t0) - (t2 - t1)}{2} \tag{2}$$

NTP assumes that the delay of both request and reply are approximately the same. If the routes do not have the same nominal delay, the synchronization has a systematic bias of half the difference between the forward and backward travel times (4).

2.2 NTPv3 within a Windows 2008 Domain

In a Windows 2008 Domain, the time is controlled by the Windows Time Service (WTS). This service uses the NTPv3 protocol to handle the time. WTS maintains several types of time synchronization(11):

- NoSync
- NTP
- NT5DS
- All

[1] Actually, there is another timestamp: the Reference Timestamp. This timestamp, however, is not used in calculating offset and delay. A description of this timestamp, as well as all other fields, can be found in appendix A

[2] t_0 = Orginate Timestamp; t_1 = Receive Timestamp (by server); t_2 = Transmit Timestamp (by server); t_3 = Receive Timestamp (by client)

The NoSync type states that the machine will use its local time as a resource and not synchronize with a time server. The NTP type stands for that the machine will use a NTP server to synchronize with. The NT5DS type is used when a machine is inside a domain. The machine will then synchronize with a reliable time server inside the domain. The All type stands for that the machine will use all the types to synchronize. It will first look for a domain time server (if the machine is a member of a domain), then it tries to contact the NTP server that is configured as peer on the machine. If neither two succeed then the machine will use its local clock to use as a time source.

Services running on a computer, inside the domain, use NetLogon to advertise its functions. NetLogon will set certain flags according to the service (like the PDC flag stands for the first Domain Controller installed in a domain). WTS makes use of these flags to inform machines inside the domain, wether the machine is a time server, or is looking for one.

Regarding WTS, Netlogon handles two different types of flags:

- TIMESERV
- GTIMESERV

The TIMESERV flag indicates that the machine is synchronized and can provide time sync responses. The GTIMESERV flag indicates that the computer is a reliable time server and that computers can synchronize with it.(12)

These two flags are represented in the registry by means of the AnnounceFlags record. This record states the mode that this machine is operating in. The available values for this record are:

0x00 Not a time server

0x01 Always a time server

0x02 Automatic time server

0x04 Always reliable time server

0x08 Automatic reliable time server

As an addition, the values can also be combined by using the bitwise OR operator.

When a client cannot receive the time from a time server it will keep trying to connect for 7 more times. When that threshold is reached, that specific time server will be “deleted” from the peer list and the client will search for another time server in the domain. If a server cannot be found, the client will not use a time server, but will stay with its local clock.(13)

2.2.1 Default configuration

Now that NTPv3 in a Windows 2008 Domain has been covered, the default configuration that is used by the Primary Domain Controller (PDC), the Domain Controllers (DCs), and members of a domain will be treated.(14)

Type The default type that WTS in a Windows 2008 Domain uses is: NT5DS. This protocol makes sure that NTP will contact the most reliable time server in the network. The election of this most reliable time server happens through the use of NetLogon, which was covered above.

Authentication WTS implements the authentication that is suggested in the NTPv3 RFC(10). The packets use a MAC code to authenticate the server that is supplying the time to the client. The process is as follows:[3] WTS implements the authentication that is suggested in the NTPv3 RFC(10). The packets use a Message Authentication Code (MAC) code to authenticate the server that is supplying the time to the client. The process is as follows(15):

1. The client sends a NTPv3 request, which contains a Key ID.
2. The server then verifies this Key ID with the local Active Directory (AD) database and checks whether it is associated with the requesting client.
3. The server responds to the request by filling in the necessary packet fields, and computing a MAC. The MAC is generated by hashing the Kerberos session key and the entire NTP packet (excluding the MAC itself).
4. After receiving the reply packet, the client also calculates the MAC in order to confirm that the server is really a domain controller within the domain and therefore trustworthy.

Discovery of other time servers The PDC of a domain will act as the most reliable time source of that domain. Any other domain controller or domain member will get their time from this PDC. The other DC(s) will also act as a time server, but will not act as a reliable time server. They are only used when the reliable time server (the PDC) can not be found.

What was discovered in the test environment, is that the DCs do not advertise automatically that they are a time server, but do allow manual discovery (by means of a specific command). This means that the clients that cannot connect to the reliable time server (the PDC) will not automatically switch to other DCs.(16)

Registery settings Looking at the AnnounceFlags registery key, the PDC will always start with a value of 0x05. This means that the domain controller always advertises as being a reliable time source.

Other DCs and members will have a value of 0x0a. This means that the DC or client will automatically determine whether it should advertise reliable time service or not. This part is connected to the discovery point made above.(17)

[3] Kerberos Authentication: http://www.diablotin.com/librairie/networking/puis/ch19_06.htm

3 Approach

The general assumption is that the attacking party has a Man-in-the-Middle (MiM) position. All network traffic will be routed from the client to server through a router, the latter being the so called middleman. From the perspective of a MiM attack there are numerous ways to interfere with the communication between a client and a server. There are two specific cases which need to be considered when forming an attack strategy:

- The NTP traffic is authenticated (i.e. signed, with a shared key).
- The NTP traffic is not authenticated.

Both these situations will be dealt with in the subsections below.

3.1 Authenticated NTP

In the first case the process of manipulating the content of the packets is significantly more difficult than in the case of no authentication. In a default AD environment the NTP packets will have two extra fields for authentication purposes which are appended to the normal packet. The first field is a 32 bit Key ID, which refers to the clients current session id. The second field is the MAC. The MAC is a hash of the session key with appended all the NTP data up until the MAC itself. Cracking the MAC in real-time is hardly a feasible option (8). But in the case of Microsoft, the request packets are not signed and therefore could be vulnerable to MiM attacks. The following methods have been looked at:

- Modifying NTP request packets.
- Modifying NTP reply packets.
- Delay NTP packets.
- Creating a DoS by flooding the server with requests requiring an authenticated reply.

3.1.1 Modifying request packets

Modifying the request packets from a client is especially of interest to an attacker, for they only indicate that the reply needs to be signed but are themselves not signed. The two extra fields Key ID and MAC are filled in by the server, but only the Key ID is used by the server. The MAC value is simply filled with zeros. By modifying some of the fields in the request packets, it might be possible to change the behaviour of the server and the reply packets the server generates.

3.1.2 Modifying reply packets

This attack focusses on how Microsoft implemented NTP authentication. It is important to know how a client behaves when receiving a modified packet. It could be the case that it discards the packet like it should, creates an event log or badly processes the modified packet as valid.

3.1.3 Delaying packets

Delaying NTP packets to create a clock skew between the server and the client is based on how NTP calculates latencies over the network. By taking the timeout value of the client NTP implementation into consideration, it could be possible to create a clock skew of a few seconds.

3.1.4 Flooding with NTP request

The theory behind this idea is that the calculation of a hash is computationally expensive, so flooding the server with NTP request would require a lot of CPU resources which might lead to a DoS. Requesting the time server is significantly less expensive, so such an attack might be quite possible. One client injecting many request could possibly lead to a DoS, but this type of attack could easily be distributed as well, which could lead to a *distributed denial-of-service* (DDoS). Although this type of attack does not aim for creating a clock skew, it does however theoretically show a possibility to deny the NTP service as a whole and therefore it is mentioned in this paper.

3.2 Non-authenticated NTP

In the second case, that of non-authenticated packets, there is no way of knowing for the server and the client what happens with the content of the packets while they are in transit. This gives a wide range of possibilities for manipulating the data in such a way that it is possible to adjust the clock on the clients side. Some of the methods that are used in the previous case are applicable here as well. Modifying non-authenticated NTP packet is not ground-breaking news. It is evident that non-signed packets can be modified, but in practice this can still be used quite often. In scenarios where non-domain clients wish to synchronize their clock with that of the PDC, they can do so by simply not using the authenticated version. The server will still obey the clients request. For this reason it is still important to mention the existence of this, rather straight-forward, attack.

4 Experiments

In this section the experiments will be described, as well as their results. First, an overview of the used test setup will be given.

4.1 Test environment

The general test environment consists of a server (PDC, time server), two clients and a router. In this subsection a general overview of this specific test environment is given, followed by a more detailed description about the various components in the test environment.

4.1.1 General Overview

The test environment has the characteristics of a normal (very) small company network. Lets first summarize the different components that the network consists of:

- 1 Windows 2008 Enterprise machine, acting as the PDC of the test domain.
- 2 Windows 2008 Standard machines, acting as two members of the test domain.
- 1 Linux Debian machine, acting as the gateway that routes traffic from the different subnets.
- 1 Switch, used to connect the different machines.

The test environment shown more clearly in Appendix B.

The three Windows 2008 machines all operate on a different subnet. As stated in the above item list, the router is used as a gateway for all these separate subnets. Between the server and the client, a switch is installed. This switch is used to route the data between the two different networks. Also a software router is installed which manages the traffic in between the server and the clients. *The various subnet configuration of the test environment is shown in Appendix B.*

4.1.2 Server

The server is installed as a Windows 2008 Enterprise machine. This server runs an Active Directory Domain called `ntptest.local`, and has the role of the PDC. As is mentioned above, the PDC is the most reliable time server in the domain and will broadcast this towards its clients and possibly other domain controllers.

4.1.3 Client 1

The first client is installed as a Windows 2008 Standard machine. We chose to manually edit the point where this client synchronizes its time with which is actually the PDC. The reason for this is in this case NTP is used instead of NT5DS, and consequently no authentication is used. With this specific machine experiments with NTP packets with no authentication enabled can be performed.

4.1.4 Client 2

The second client is installed as a Windows 2008 Standard machine. The configuration on this client was purposely kept default. Like stated in section 2, the default configuration of a Windows machine in a domain is that it uses the WTS type: NT5DS. This means that Kerberos will be used for the authentication of the time server. Client 2 will be used for experiments that require enabled authentication.

4.1.5 Router

The router is installed as a Debian Squeeze machine, configured to act as a router for the various subnets in the test environment. It also serves as the machine that will perform the various attacks against NTP.

For performing a MiM attack a specific tool was written[4] for capturing, manipulating and injecting NTP packets. It will automatically add rules to netfilter for blocking original NTP traffic.

4.2 Authenticated NTP

In this subsection, the experiments regarding authenticated NTP packets will be described. Along with each attack strategy the corresponding results are presented.

In order to measure the clock skew between two systems, a tool[5] was written for Windows Powershell. The tool sends every specific time interval a time request to both systems, calculates the clock skew, and logs it to a file. The obtained files were visualized in the figures below.

4.2.1 Modifying request packets

The following experiments have been performed to test whether the specific time protocol was properly implemented:

- Removing the *authenticator* part from the request packet.
- Changing the Leap Indicator field.

[4] `https://www.os3.nl/_media/2012-2013/students/dennis_pellikaan/ssn/ntp.tar.gz`
[5] `https://www.os3.nl/_media/2012-2013/students/tjebbe_vlieg/ssn/time.ps1.gz`

- Changing the Poll Interval Value.

After experimenting with changing fields that could influence the time on the client side, there appears to be only one field in the request header that can be used for influencing the outcome of the response packet. The Poll Interval Value is used by the client to adjust the time to wait before the next NTP request is sent. This values may lie in the range of 4 - 17 and will be calculated to seconds with the formula 2^n, where n is the Poll Interval Value. Normally these values depend on how often the clients clock has been correctly synchronized, but if the client places a non-default value here, then the server will simply copy that value into the response packet. In the two extreme cases it is possible to let the let client automatically do an NTP request every 16 (2^4) seconds or 131072 (2^{17}) seconds. Although this may be seen as feature or a weak vulnerability. The fact that is possible to do these kinds of modifications should not exist. These kind of problems could have easily be eliminated by authenticating the request packets as well.

4.2.2 Modifying reply packets

The following modifications to the reply packets have been performed to test the NTP implementation:

- Changing each individual field in the NTP header. See figure 6 in appendix A for all available fields. It appears that the server indeed does use each individual bit for creating the MAC. Therefore the client seems to correctly validate the MAC with the NTP packet, making it is not susceptible to these types of attacks.
- Removing the authenticator part from the reply packet.
- Changing the MAC value with arbitrary data.
- Setting the MAC value to all zeros.

All of these approaches resulted in the client rejecting the reply packet. The modifications are thus correctly detected by the client and the reply packets are discarded, making modifying of the reply packet an infeasible operation.

4.2.3 Delaying packets

As was pointed out in subsection 3.1.3, the goal of the delay attack is to create a small clock skew by delaying NTP requests, NTP replies, or both. The hypothesis is that a one-way delay (either the request or reply) will cause a clock skew of half the delaying time, whereas delaying in both directions (both request and reply) will cause no skew, as in that case the NTP client will successfully be able to calculate the round-trip-delay. To test these hypotheses, two tests were performed: In the first test all replies from server to client were delayed for 4 seconds. In the second test both requests from the client and replies from the

server were delayed for 2 seconds. It was not possible to enforce a delay of more than 4 seconds for in that situation the client would time out.

During the first experiment, a clock skew of two seconds was achieved in an interval of twenty minutes, which was as expected. The resulting graph can be seen in figure 2. It is clearly visible that Windows adjusts its local time in small steps of time. The reason for the clock skew being two seconds instead of the 4 seconds of the enforced delay can be explained by the way NTP calculates time offset and delay. As was explained in section 2 and especially equation 2, the NTP client assumes the measured delay to be spread equally over the round-trip of both request and reply. In this case, this assumption is violated by the middleman, who causes the reply to have a substantially longer delay than the request has. The bogus time offset can be calculated as follows (θ meaning offset; σ meaning skew; d meaning malicious delay):

$$\begin{aligned} \theta &= \frac{(t_1 - t_0) + (t_2 - t_3)}{2} \\ \theta^* &= \frac{(t_1 - t_0) + (t_2 - (t_3 + d))}{2} \\ \sigma &= \theta^* - \theta = \frac{-d}{2} \end{aligned} \tag{3}$$

It is interesting to see that the graph representing the time adjustment is not a straight line, but has the appearance of a polynomial function. The time adjustment mechanism is not a property of NTP, but of the specific operating system. Evidently, Windows Server 2008 begins correcting a clock skew large(ish) steps and decreases this step size over time, up until the time correction has been processed.

It can further be seen that there is some jitter in the measurement points. This is probably partly because of measurement error: packets sent by the powershell script also have round-trip delay which cause measurement error. Another reason for this jitter is the clock error of the local clock of the measured systems.

In the second experiment both NTP requests and replies were delayed for the same amount of time (in this case two seconds). The results, which can be seen in figure 3 show no significant clock skew in between server and client, according to the hypothesis. The lack of clock skew can also mathematically be explained (again θ meaning offset; σ meaning skew; d meaning malicious delay):

$$\begin{aligned} \theta &= \frac{(t_1 - t_0) + (t_2 - t_3)}{2} \\ \theta^* &= \frac{((t_1 + d) - t_0) + ((t_2 + d) - (t_3 + 2d))}{2} \\ &= \frac{(t_1 - t_0) + (t_2 - t_3) + 2d - 2d}{2} = \theta \\ \sigma &= \theta^* - \theta = 0 \end{aligned} \tag{4}$$

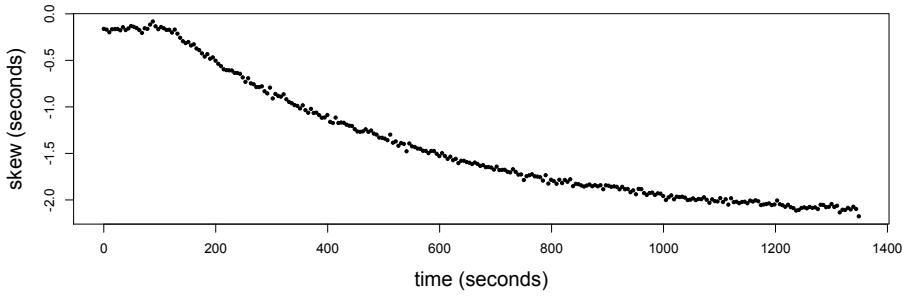

Fig. 2: Delay attack: 4 seconds delay

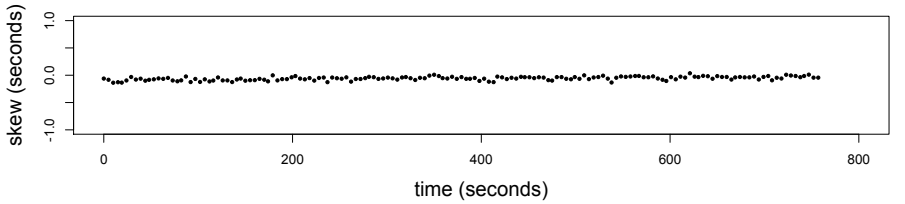

Fig. 3: Delay attack: 2 seconds delay, both directions

4.2.4 Flooding the server with NTP requests

Generating a MAC requires some processing power of the PDC. By intercepting an genuine NTP request packet makes it possible to flood the server by replaying that specific packet in large amounts and increasing the server's CPU load rapidly. There is no method preventing this attack in the current implementation (9). It is of course possible to create a new NTP request from scratch, but in that case a valid Key ID has to be guessed. From a MiM position an intercepted Key ID can be used. The following pseudo-code shows the basic principle of the implementation that was used to perform this attack.

```
capture request packet
packet.eth_src = ROUTER_ETH_SRC
loop infinitly
    inject packet
```

Measuring the server's CPU load points out that all four cores are under almost a maximum load of 100%, whilst the router injecting the packets its CPU load peaks no higher than 8% for all cores combined.

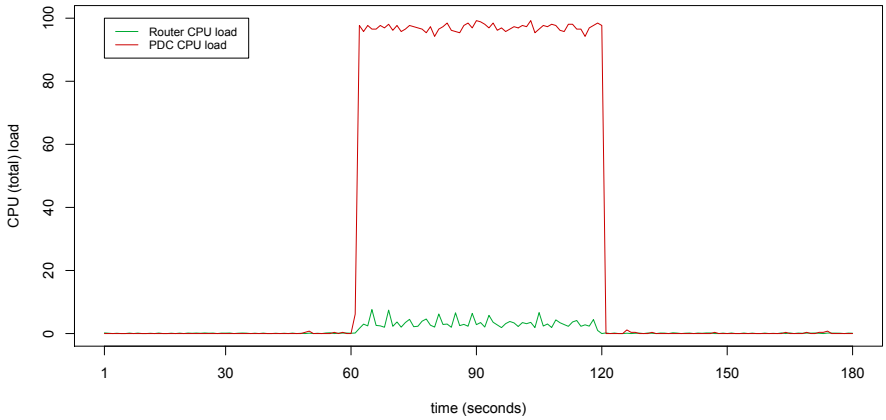

Fig. 4: CPU load during DoS attack

As shown in figure 4 it is quit easy to increase the CPU load to its maximum. During the experiments the system showed some instabilities, but because of the limited time-span for this project, there was not enough data available to proof that this was solely caused by the DoS attack.

4.3 Non-authenticated NTP

Although it is widely known that non-authenticated NTP is vulnerable to MiM attacks, it is definitely still worth mentioning for there are still many systems that do not use authentication. There are multiple reasons for a system not using authentication, for example:

- **Non-interoperability**
  Different operating systems that want to use the same time source, without making to many modifations to their local system.

- **External time source**
  Some administrators prefer to use a time source that lies outside of their domain. Most often their has not been a key exchange with the external machine and therefore use non-authenticated NTP.

In combination with the Poll Interval Value modification, modifying NTP replies is a very interesting method to rapidly create a significant clock skew on a clients machine, without greatly increasing or decreasing the timestamp values. Figure 5 shows a clock skew of five minutes created in a period of less than 5,000 seconds by letting the client poll the new time for every 16 seconds.

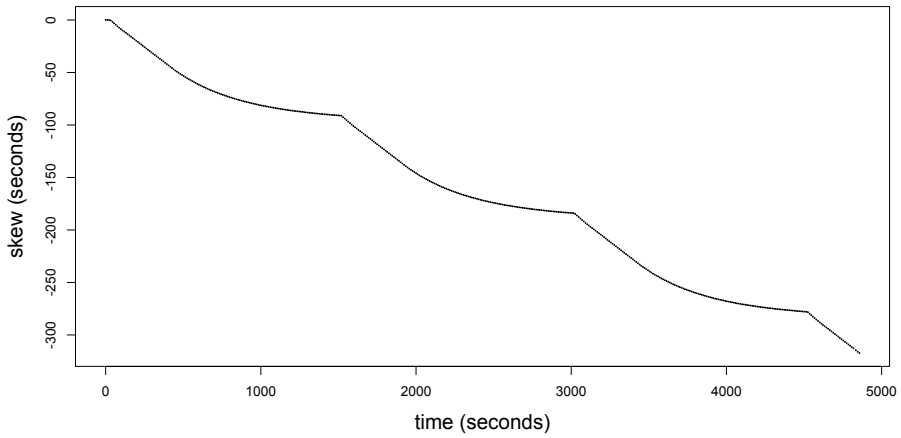

Fig. 5: Modified reply packets: 300 seconds delay

It is interesting to see that there is a wave-like pattern in the graph, which is probably caused by the operating system first succeeding a single time adjustment, before processing another adjustment. All these adjustments have the same shape as the one treated in subsection 4.2.3.

5 Conclusion

Our research question was whether it would be possible to create a significant clock skew between two systems in a Windows Active Directory environment with a default configuration. As was pointed out in subsection 2.2, Windows Active Directory's default implementation of NTP does use authentication by means of a MAC code. This MAC code makes it impossible to modify any of the included fields in the NTP response packet. The only way to be able to modify these fields is to somehow obtain the used session key, which might not strictly be impossible, but achieving this would enable much more interesting attacks than creating clock skews and surely goes out of the scope of this article.

Although modifying NTP replies is practically impossible when a MAC is present, situations are possible in which this kind of authentication is not enabled. As has been stated in subsection 4.1.3, using NTP instead of NT5DS disables authentication by default. This could be the case in a network with multiple operating systems. Another situation in which no authentication is present is when a system synchronizes with an external time source.

The experiments can be divided into two groups: the ones which apply to non-authenticated packets and the ones which are also possible with authenticated packets. Evidently, creating a clock skew in a system with no authentication enabled is much simpler to achieve than in one that has authentication enabled, what would authentication be useful for otherwise? For this reason, much greater clock skews in much less time can be created in non-authenticated systems.

With non-authenticated packets it is simply possible to modify NTP replies, altering the time every request. This approach leads to a significant clock skew within reasonable time: in the particular experiment a clock skew of five minutes and 17 seconds was achieved in eighty minutes and 57 seconds. Not really surprisingly, it can thus be concluded that not having authentication enabled within a network is very unsafe, as it malicious parties could easily alter the time on specific systems.

In the case that authentication *is* enabled, these kinds of modifications are not possible. That is, only the response packet is signed with a MAC, so theoretically the request packet *can* be modified, as in these packets no MAC was present. In most cases however, this still is not possible, as either these modifiable fields would not result in a clock skew, or the modifications would be detected by the client after receiving the reply packet. There was however one specific field that could be adjusted modifying the behaviour of the client, namely the Poll Interval field. This field can be adjusted to any value in the range 4-17, leading to a new poll interval in the range 16 seconds to 36 hours. Although this attack does not immediately create a large clock skew, it is clear to see that a third party should not be able to change the behaviour of a system's time synchronization service. This weakness could easily be solved by making the client sign its packets as well.

Another feasible attack is the delay attack. This specific attack exploits NTP's assumption that a packet's delay is equally spread over the entire round-

trip. The experiments, in which a reply packet was delayed for four seconds, enforced a clock skew of two seconds. Obtaining a larger skew was not possible due to time-outs of the client's time process. The delay attack does not change any value in the packets sent, so no form of authentication can prevent this kind of attack. As the clock skew that can be obtained by this attack is only minimal, the delay attack is not that much of a threat to systems.

A final interesting attack strategy, which does not necessarily fit into the research question of this article, but is interesting nonetheless, is a denial-of-service attack by means of flooding the server with replayed NTP requests. An important notion to make is that the sent requests should request authenticated response packets, for the calculation of these MACs is computationally heavy for the server, which causes the denial-of-service. The provided experiments show that with a reasonable amount of computing power of the malicious machine, a time server's CPU can be loaded to almost 100 percent.

The main conclusion is thus that time synchronization in a Microsoft Active Directory system, running on Windows Server 2008 systems, is quite safe. That is, it is only safe if authentication by means of a MAC field is enabled. If such authentication is not enabled, the system could easily be the target of attacks which rely on clock skews, for in that case, it is almost straightforward to enforce clock skews.

References

[1] Kopka Helmut, W. Daly Patrick, *Guide to LaTeX*, 4^{th} Edition. Available at http://www.amazon.com.

[2] Graetzer George, *Math Into LaTeX*, Birkham user Boston; 3 edition (June 22, 2000).

[3] J. Postel, User Datagram Protocol, *RFC 768*, August 1980, available at http://www.ietf.org/rfc/rfc0768.txt.

[4] Gotoh, T; Imamura, K; Kaneko, *Improvement of NTP time offset under the asymmetric network with double packets method*. Conference on Precision Electromagnetic Measurements, 2002. pp. 448–449.

[5] R. Braden,D. Borman , Computing the Internet Checksum , *RFC 1071*, September 1988, available at http://tools.ietf.org/rfc/rfc1071.txt.

[6] Mills, David L.*Computer Network Time Synchronization: The Network Time Protocol on Earth and in Space, Second Edition*. CRC Press, Inc., 2010

[7] Matt Bishop. A Security Analysis of the NTP Protocol, 1990.

[8] Thomas Peyrin and Yu Sasaki and Lei Wang, Generic Related-key Attacks for HMAC, 2012

[9] B. Haberman, Ed., D. Mills , Network Time Protocol Version 4: Autokey Specification, RFC 5906, June 2010, available at http://tools.ietf.org/html/rfc5906.

[10] Network Time Protocol (Version 3) Specification, Implementation and Analysis http://tools.ietf.org/html/rfc1305

[11] Types of peer synchronization:
http://technet.microsoft.com/en-us/library/cc779145(v=ws.10).aspx

[12] How the Windows Time Service works:
http://technet.microsoft.com/en-us/library/cc773013

[13] Time Server Election in a Windows Domain:
http://blogs.msdn.com/b/w32time/archive/2008/05/29/
to-be-reliable-or-not-to-be-reliable.aspx

[14] Configuration Options of Windows Time Service:
http://technet.microsoft.com/en-us/library/cc773263(v=ws.10).aspx

[15] Kerberos Authentication:
http://www.diablotin.com/librairie/networking/puis/ch19_06.htm

[16] Discovery of other Time Servers:
http://technet.microsoft.com/en-us/library/cc756494(v=ws.10).aspx

[17] Standard Registery Values:
http://msdn.microsoft.com/en-us/library/cc246923(v=prot.20).aspx

A NTP Packet

In this appendix a detailed description of the NTP packet is given. Figure 6 shows the syntax of the NTP header, of which the various fields are explained below.(10)

Leap Indicator This two-bit integer specifies whether a leap second is to be inserted in the NTP timescale. 23:59:59 is followed by 23:59:60, after which the next day starts with 0:00.

Version Number This field of three bits indicates the NTP version that is currently used.

Mode The mode field (6 bits) is used to indicate in what mode the sender of the packet is. 7 is reserved for private use, 6 is reserved for NTP control messages, 5 is used for broadcast mode, 4 is the server mode, 3 client mode, 2 symmetric passive, 1 symmetric active and 0 is reserved.

Stratum This integer indicates the stratum of the local clock.

Poll Interval This signed integer (8 bits) indicates the minimum interval in seconds between NTP requests, as a power of two.

Precision This signed integer indicates the precision of the local clock of the packet's sender, in seconds to the nearest power of two.

Root Delay This signed fixed-point number (32 bits) indicates the total round-trip delay to the primary reference source at the root of the synchronization subnet.

Root Dispersion A signed fixed-point number (32 bits) indicating the maximum error relative to the primary reference source at the root of the synchronization subnet

Reference Clock Identifier This 32-bit field identifies the particular reference clock. When it is a stratum 0 or 1 reference, a four-octet, left-justified, zero-padded ASCII string is used.

Reference Timestamp This is a timestamp (64 bits) representing the last time the local clock of the packet's sender has been updated. These value is zero if the specific system has not yet been synchronized.

Originate Timestamp This timestamp represents the local time at which the NTP request was transmitted.

Receive Timestamp The local time of the time source, at the time of receiving the NTP request.

Transmit Timestamp The local time at the sender of the packet at which the packet was sent in timestamp format, written in timestamp format. In the NTP reply, this is the local time of the time source at the moment that the reply was sent.

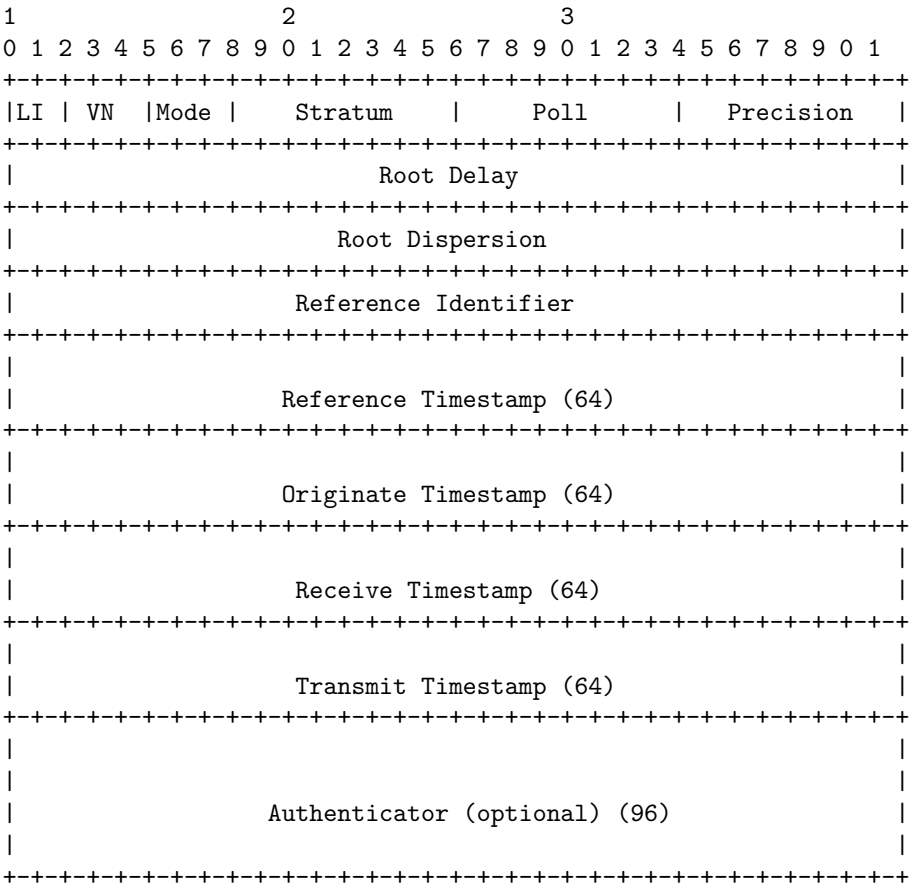

Fig. 6: NTP packet header

B Technical details

In this appendix a technical overview of the test environment in which the experiments have been performed, is given. Figure 7 illustrates how the network was set up, whereas table 2 specifies the used hardware in detail.

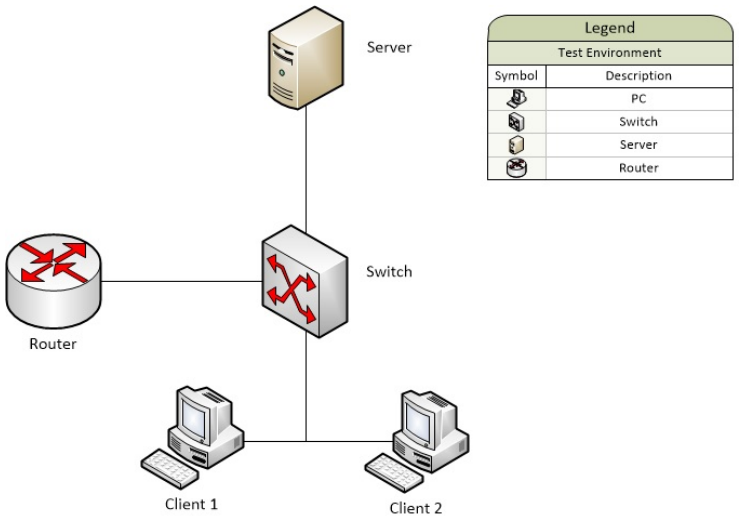

Fig. 7: Network Diagram

	Interface	IP adres	Mask	Gateway
Router	Eth0	192.168.1.254	255.255.255.0	
	Eth0:1	192.168.2.254	255.255.255.0	
	Eth0:2	192.168.3.254	255.255.255.0	
	Eth1	145.100.104.61	255.255.255.224	145.100.104.1
Server	NIC 1	192.168.1.1	255.255.255.0	192.168.1.254
Client 1	NIC 1	192.168.2.1	255.255.255.0	192.168.2.254
Client 2	NIC 1	192.168.3.1	255.255.255.0	192.168.3.254

Tab. 1: Details network environment

Hardware	Dell Poweredge R210 II
Role	PDC
Name	NTPserver (Wakefield)
Operating system	Windows 2008 Standard 32 bit SP1
Processor	Intel Xeon CPU E3-1220L v2 - 2,30 Ghz
Memory	8 GB
Harddisk	Seagate ST1000NM0011 - 1TB
Network	Broadcom BCM5716C NetXtreme II GigE
Hardware	Dell Poweredge R210 II
Role	Domain client 1
Name	NTPclient 1 (Sheffield)
Operating system	Windows 2008 Standard 32 bit SP1
Processor	Intel Xeon CPU E3-1220L v2 - 2,30 Ghz
Memory	8 GB
Harddisk	Seagate ST1000NM0011 - 1TB
Network	Broadcom BCM5716C NetXtreme II GigE
Hardware	Dell Poweredge R210
Role	Domain client 2
Name	NTPclient 2 (Helsinki)
Operating system	Windows 2008 Standard 32 bit SP1
Processor	Intel Xeon CPU L3426 - 1,87 Ghz
Memory	8 GB
Harddisk	WDC WD500 GB
Network	Broadcom BCM5716C NetXtreme II GigE
Hardware	Dell Poweredge R210
Role	Router
Name	Router (Vienna)
Operating system	Debian Squeeze 64 bit
Processor	Intel Xeon CPU L3426 - 1,87 Ghz
Memory	8 GB
Harddisk	WDC WD500 GB
Network	Broadcom BCM5716C NetXtreme II GigE
Role	Switch
Hardware	Cisco 2950
Ports	24 ports

Tab. 2: Hardware overview

C Acronyms

The following table provides a collection of common abbreviations used in this research paper:

DC	Domain Controller
DDoS	Distributed Denial of Service
DoS	Denial of Service
MAC	Message Authentication Code
MiM	Man in the Middle
NTP	Network Time Protocol
PDC	Primary Domain Controller
WTS	Windows Time Service