Preventing Pass-the-Hash and Similar Impersonation Attacks in Enterprise Infrastructures

Alexander Oberle, Pedro Larbig, Ronald Marx,
Frank G. Weber, Dirk Scheuermann
Fraunhofer Institute for Secure Information Technology (SIT)
Rheinstraße 75, 64295 Darmstadt, Germany
{oberle|larbig|weber|scheuermann}@sit.fraunhofer.de,
ronald.marx@gmx.de

Daniel Fages, Fabien Thomas
Arkoon Netasq
1 Place Verrazzano, CS 30603 69258,
Lyon Cedex 09, France
{daniel.fages|fabien.thomas}@stormshield.eu

***Abstract*—Industrial espionage through complex cyber attacks such as Advanced Persistent Threats (APT) is an increasing risk in any business segment. Combining any available attack vector professional attackers infiltrate their targets progressively, e.g. through combining social engineering with technical hacking. The most relevant targets of APT are internal enterprise and production networks, providing access to top-secret information. This work focuses on preventing *Pass-the-Hash*, one of the biggest and most long-standing security flaws present in enterprise domain networks. The introduced approach can be applied further to make password theft pointless for attackers in general, and is capable of extending network protocols, that are unprotected by themselves, with approved security mechanisms. The protocols do not need to be modified and already existing network services can stay untouched when integrating the solution into enterprise infrastructures.**

***Index Terms*—Enterprise security, single sign-on, APT, Pass-the-Hash, Pass-the-Ticket, vulnerability, password theft, authentication, mitigation, prevention, protection, proof of identity**

I. Introduction

Economic and industrial espionage is one of the biggest threats for companies, organizations, public institutions and big players. In the ISMG's recent APT survey of 2014 [1], *"22 percent of companies know they experienced a security incident as a result of APT in the past year"*. The survey goes on to say that *"53 percent give themselves a C or worse when assessing their ability to defend against APT"*. As one conclusion the survey argues that in reality more than every fourth company should be affected, the majority just does not know.

With respect to APT, attackers prepare the infiltration of espionage targets well in advance. This type of attack includes the manipulation of inter-personal relationships (social engineering), for example to gain access to login passwords. Particularly coveted goals are internal networks that transport and process company data and secrets, as well as production and industrial plants. Information from local networks is valuable. Typically, access control protects local networks against the outside, while often completely neglecting internal network protection. If an attacker succeeds in gaining access to a network, or is already part of it (infiltration by insiders), he can possibly collect sensitive data such as internal secrets or passwords for lateral movement and thus can gain further access, or he might even be able to take down the network from a single entry point and cause widespread economic damage.

This work focuses on preventing one of the most popular attacks, called Pass-the-Hash (PtH) [2], which has been used for lateral movement and data exfiltration in Microsoft (MS) domain networks for more than 15 years [3]. The approach to be introduced will address the PtH problem at its root so that lateral movement and gathering access to sensitive information will be consequently prevented in the network.

Chapter II introduces the background of this work and PtH at a glance. The solution to prevent PtH network attacks is subsequently presented in Chapter III. With focus on usability the integration into enterprise networks is given in Chapter IV. Summarized in Chapter V, a proof of concept is integrated into a network appliance that is utilized to evaluate a first application-related prototype. Chapter VI concludes this applied research work and gives an overview about the necessary next steps to establish a flexible solution in practice.

II. Background

Existing single sign-on (SSO) authentication mechanisms in MS Windows are lacking in conceptual identity protection so that stolen credentials can be reused [2], [3]. When a user logs into his client computer that is connected to a Windows-based network domain he typically has to enter his user name and password. After successful authentication towards the Domain Controller (DC), a hash value from the password is stored locally in the client's memory. In case that the user requests a service involving a remote network resource (such as a file server), the client sends an access request to the respective resource. In order to verify the user authentication for accessing the resource, the server forwards a login challenge request to the client. The client solves the challenge by using its saved password hash. Upon receipt, the server forwards both the original challenge and the client's response to the DC, which verifies the response to the challenge and the related password hash used by the client. In the case that the hash and the user's password hash match, the DC notifies the server which in turn

grants access to the client. The client's password hash itself has not been available to the server and is not stored there throughout the basic authentication process.

The basic challenge that exists in Windows domain networks is the possibility to reuse stolen passwords or password hashes and thus taking over the identity of a valid user. Related to SSO domain networks, this threat is known as *Pass-the-Hash*. PtH is still the first choice for espionage and lateral movement in most enterprise networks for the following reasons. The password hashes in MS Windows domain networks are not tied to an authentication request by design, thus an attacker can continuously re-use any stolen credentials. This is somewhat mitigated by using Kerberos Tickets which expire after a certain time, but this does merely force an attacker to act quicker, usually in a matter of several hours.

A. Pass-the-Hash at a glance

Pass-the-Hash attempts are usually based on injecting stolen credentials into the memory of an attacker controlled computer. These credentials are usually the user name and the hash value of the corresponding password or Kerberos Tickets that were gathered illegally beforehand [2], [3]. Afterwards, the normal service authentication mechanism takes place using the injected credentials. The goal of a PtH attack is to obtain access to a network resource (such as a file share) in MS Windows domains without being authorized. This attack can remain undetectable when accessing network resources until the original password is changed by the user or administrator since no invalid logins occur. PtH attacks are typically accomplished by taking advantage of the following circumstances and facts:

- Theft of valid credentials (password hash, user name, ticket), e.g. through malware and memory dumps,
- Access to an appropriate network component in the network infrastructure of the attack target (no SSO authentication needed beforehand),
- Ability to inject or otherwise use stolen credentials in the service authentication process towards network resources.

B. Windows smartcard authentication

The problem of misusing a password or password hash is often addressed by using a second factor when authenticating. To avoid any misunderstanding, these *Two Factor authentication* mechanisms (e.g. by smartcard or SMS) are generally only applied to the initial DC domain SSO authentication, missing the 2^{nd} Factor to be further evidenced when authenticating towards services and servers in DC-managed domain networks. The following section briefly introduces the basic Windows smartcard Two Factor authentication to resolve the erroneous belief of being able to mitigate the PtH vulnerability.

Microsoft Windows domain authentication innately supports smartcard tokens as an additional factor in authentication. In Windows domains, smartcard based login mechanisms may be configured in a number of different ways. Standard Windows login systems use smartcard authentication as a direct alternative to password-based logins, i.e. the user is authenticated as soon as the smartcard successfully authenticates against the DC. The smartcard proves to the DC that it is aware of its private key and that the smartcard is actually present. As a result, a password hash, which is validated in further service authentications against the DC's database, is stored in the client's memory once again. If the password authentication is disabled in the default domain policy and there is no password hash stored on the DC, a unique random number is used for each client. Note that this random number can also be misused by an attacker in the same way, since it is not updated on subsequent logins and never expires by default. Likewise, password-based and smartcard-based procedures can be used for login, but replaying already gathered password hashes remains a possible attack risk, since the hashes will still be accepted for proof of authentication for services and servers. In summary, the default smartcard authentication does not provide an advantage over password-based login for mitigating or even preventing PtH attacks.

III. Solution design

In order to mitigate the PtH vulnerability, a *correlating* 2^{nd} *Factor token authentication* is designed and introduced in the following. The additional authentication can be applied to any vulnerable authentication mechanism in general, or even to protocols without authentication at all. There only has to be a possibility to identify the requesting party that is trying to access. If authentication is given, it is essential to identify the client while the authentication session takes place and before any other sensitive data is transferred. Access to the target service must then only be allowed when the extracted identity matches the one of the 2^{nd} Factor authentication, which must have been successfully validated first.

A. Generalised application & definitions

The mitigation scenario is defined to enhance the security level when the client starts to authenticate towards a server or service, further called SRV AuthN. This is at the time the identity information (legitimate or not) is already in use by a client, thus after the usual SSO domain authentication.

Identification & Correlation: The most important aspect on the server side is to identify the client while the original SRV AuthN takes place, and to subsequently *correlate* his identity that is used to access the service with the extended 2^{nd} Factor token authentication. The identity of the SRV AuthN is further called *Requester Identity* (RID). The approach to be introduced, which is controlling the entire improved service access, is further called *Correlating Authentication Service* (CAS). The RID of the SRV AuthN is identified by the CAS and bound to the 2^{nd} Factor token authentication, further called CAS AuthN.

Basic requirement for client & server: The CAS facing the client must be verifiable (e.g. by using certificates). The client must be assignable to a unique RID and be provided with the private token (matching its unique RID) so that it is able to

fulfil the additional CAS AuthN. The CAS therefore needs the public verification data correlated to the RID and the possibility to identify the client explicitly by its RID before challenging the client to evidence the presence of its private token.

Objective and benefit: Even if the attacker owns a legitimate token, it is unusable with stolen login information not matching the respective 2^{nd} Factor, CAS AuthN token. Thus, the introduced solution even prevents insider attacks.

B. Correlating Authentication Service

This section gives an abstract overview on how the CAS that is controlling SRV and CAS AuthN has to be designed to prevent the PtH replay attack successfully.

Sequence chart 1 illustrates the fundamental integration as well as the different parties: client and service, the original service authentication (SRV AuthN), the CAS as a superordinate service within its additional token authentication, CAS AuthN. The existing SSO domain authentication does not affect the introduced solution and is not shown in Figure 1. Furthermore it is assumed that the client side is already in possession of the password hash that is needed to authenticate towards the untouched SRV AuthN and related services, and that the identity of the CAS has already been verified successfully, for example by using server certificates when establishing the connection to it. The CAS is responsible for controlling the entire authentication process as follows:

1. A service access request (AREQ) is sent by the client.
2. When this request is received by CAS, the RID will be extracted from the AREQ.
3. CAS generates a fresh, unique challenge.
4. CAS is challenging the client.
5. The client solves the challenge thus evidencing the presence of its CAS AuthN token, which is correlated on the CAS to the RID.
6. The client sends the challenge solution to CAS.
7. CAS verifies the client's evidence of the CAS AuthN token's presence, correlated to the extracted RID. Verify_token$_{RID}$ might be equal to auth_token$_{RID}$ (cf. step 5) in case symmetric cryptography is used.
8. After successful CAS AuthN, the access request AREQ of step 1 is forwarded from CAS to the SRV AuthN back end.
9. The resulting ARESP of the SRV AuthN is returned from the back end to CAS.
10. The ARESP of the SRV AuthN is forwarded from CAS to the client.
11. After successful SRV AuthN, confirmed by a valid ARESP, the client sends data requests REQ.
12. CAS ensures that this data request is forwarded only if CAS AuthN and SRV AuthN succeed as well, on the basis of the RID as described before.
13. The service processes the REQ.
14. The response is forwarded to CAS.
15. CAS forwards the response to the client and the service is established.

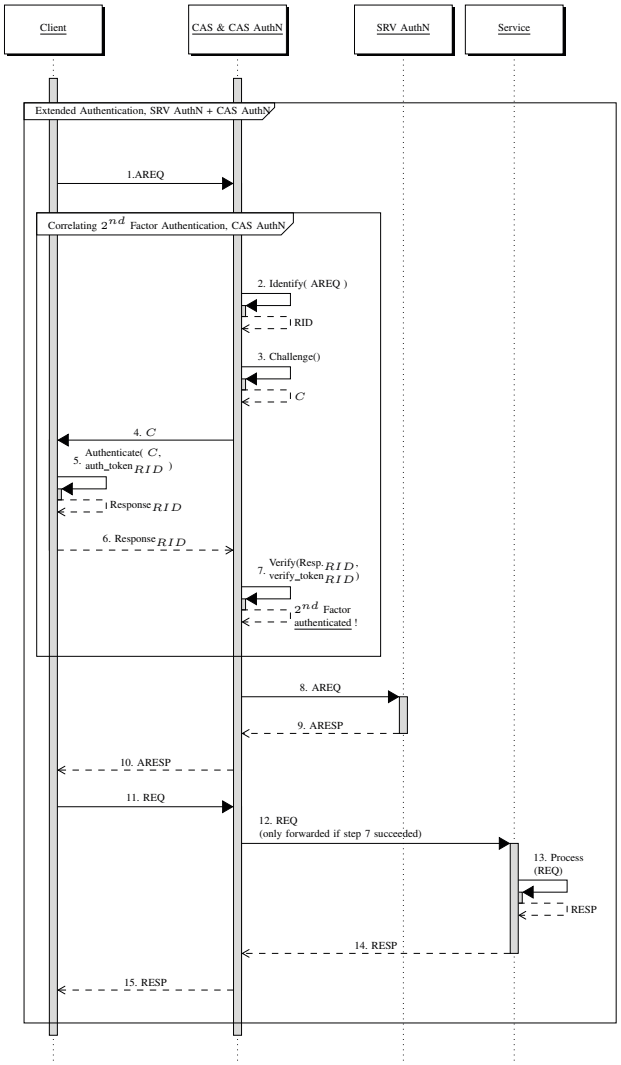

Fig. 1. CAS-secured service authentication

Notes on the original SRV AuthN: The existing SRV AuthN is initiated by the client in step 1, Figure 1. The unmodified but controlled steps of SRV AuthN, cf. step 8 to 10, can be processed in parallel with CAS AuthN. Thus these steps may take place before, during, or after the CAS AuthN process. It is vital that both, the SRV and CAS AuthN processes finish successfully before granting service requests any access to the service after step 11 et seqq. It is therefore essential that the superordinate CAS delays or blocks any access that may read or write non-authentication data on the target service before the full correlating authentication process is completed. To give an attacker the lowest possible surface to interact with a possibly insecure service authentication (SRV AuthN), it is recommended to process the CAS AuthN before the existing SRV AuthN as shown in Figure 1.

IV. Integration of CAS into enterprise networks

A CAS can be realized in different ways. A brief introduction will be given in the next sections. The crucial parts will be discussed and compared to each other and a generalized solution will be introduced in the form of an abstract as well.

A. CAS as server side extension vs. network appliance

Two main focuses on how to realize the solution practically emerge: a decentralized and a centralized approach. If choosing to extend the server side, it is possible to integrate a CAS as a kind of a fire walling service directly on the affected target server (CAS-S). In contrast, the application of an intermediary CAS appliance (CAS-A/V) introduces a centralized solution instead of an extension on each service host. Realizing the centralized CAS-A/V alternative results in additional requirements to form a complete secure infrastructure, but server side modification is no longer required.

Additional requirements on the centralized approach: The CAS-A/V centralized integration into the infrastructure results in additional security policy requirements related to the infrastructure. Depending on services and components of an infrastructure, the resulting security policies may have to be designed individually, but they can be outlined as a fundamental security requirement. To reach a security level that is comparable to the decentralized approach, the following fundamental policies must be considered:

- CAS-A/V has to be verifiable by any client in the network
- Accurate network separation between client and the secure service zone, which is hosting the CAS-A/V protected services and servers
- No possibility for lateral movement inside the protected service zone (e.g. disabling remote execution services, disallowing communication between services)
- Protection mechanisms against attacks beyond PtH like Man-in-the-middle (MITM) and address spoofing must be in place
- Redundancy to avoid a single point of failure and for optimizing performance in large scale networks

Obviously, several security policies have to be applied to protect the resulting infrastructure against further threats. This speaks for the decentralized approach (CAS-S) which is easier to protect but harder to integrate into an existing network.

B. Securing beyond PtH

Confidentiality and integrity protection is needed that will prevent attackers from falsifying messages or taking over sessions of the underlying protocols, and thus protects against attacks that may take place beyond PtH attempts. Additionally required protection against MITM attacks can be implemented by separating the networks virtually. It may be accomplished by routing the connections via the CAS over an IPsec/TLS connection [4], [5], [6] or VPN-based tunnels to the secure service zone as well. These tunnels can further be used to establish a CAS AuthN directly, e.g. by a connection establishment with smartcard authentication.

Thus, the vulnerable SRV AuthN is executed inside the established secure channel and the identified RID is correlated to the transporting tunnel that was established securely beforehand by the user's token. This extended approach covers bright prospects and is presumed to be the recommended solution. Without this additional security measure, only the authentication mechanism is secure while other vulnerabilities may still exist on underlying session and transport layers.

C. Choice of tokens

The choice of tokens can be matched to the infrastructural requirement. The CAS AuthN token is recommended to be backed by a hardware token such as a *smartcard* (e.g. RSA [7] based Windows compatible PKI cards, or similar public key cryptography tokens), which must be present at the requesting machine during the normal (and possibly vulnerable) authentication cycle. Note that losing hardware tokens can be detected quickly and creating copies of them is not trivial, whereas a password theft usually takes place invisibly. If tokens such as smartcards are impractical, any other identification that is hard to duplicate can be used, like USB security tokens, biometric scanners etc. Those tokens can be directly integrated in the hardware of a network component, so that the private key is protected without any need of user interaction.

Periodical attestation of token presence: It is recommended that the presence of the client's CAS AuthN token is proven periodically towards the CAS. This time intervals may be configurable at the CAS for individual use cases. Since the challenge to be signed by the client's token is newly generated, thereby re-authenticating the client, the CAS directly verifies the continued physical presence of the client's token while processing the protected services at the same time. When using the recommended smartcard solution, the user PIN unlocks the smartcard at the initial SSO login, so that no further user interaction is needed, but the physical presence of the smartcard.

Machine-to-Machine: The approach stays compatible with Machine-to-Machine (M2M) interaction, since the most important aspect is to protect the private key, so that the authentication data for CAS AuthN can not be stolen from the participating machine. If pluggable tokens such as smartcards cannot be applied securely to M2M scenarios (e.g. unlocked rooms and the risk of anyone stealing the token that is permanently connected to the machine), integrated hardware chips and trust anchors may be taken as a basis for protecting the private key for CAS AuthN. To manage high security areas, additional concepts can be combined to the introduced approach in this work. Using an integrated hardware module like a Trusted Platform Module, mechanisms to prove the trustworthiness of the client's operating system can be used to augment the CAS functionality, cf. *Trust Establishment and Authentication, TEA Protocol* [8].

D. Multi-user and identity management

The CAS solution can be very flexible when managing user access and multiple accounts. For example this approach can allow administrators, who have multiple accounts, to authenticate themselves with a single smartcard for both their

standard user account for day-to-day work and their privileged account for administrative work. Furthermore, group accounts (e.g. admin@admin, cf. Figure 2) can be associated to and shared amongst multiple smartcards, which has the advantage of being able to identify the real users by their smartcard identity. This also allows to easily integrate new credentials once a smartcard certificate expires: The Admin can simply mark both tokens as valid for the transition period.

In addition, it allows to detect insider attacks. If a legitimate user's workstation, that has already established CAS AuthN, is trying to execute a PtH attack by injecting stolen credentials into a protected session, the access is denied and the attack source can be identified by the smartcard used. Without a valid smartcard an attacker is not even able to establish a communication channel towards a CAS and subsequently to a target service. An example of possible identity management is illustrated in Figure 2.

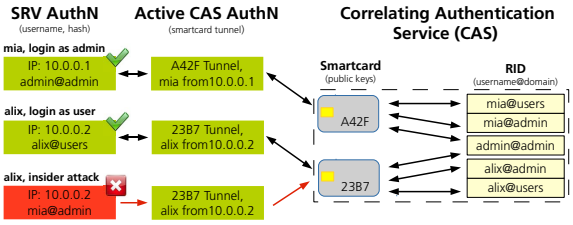

Fig. 2. Exemplification and benefits of the CAS identity management

E. Operating system integration and usability aspects

In order to create an easily usable security solution it shall not require any undesired or incomprehensible tasks from a user. The presented approach can be integrated into an operating system's existing login mechanisms, so that the users cannot even detect it's presence in day to day use. The only difference from an unprotected environment is the fact that access is impossible when the token is being removed from the machine.

An additional benefit from this solution is that users will regain control of their credentials and can invalidate all their sessions by simply removing the token in case of a suspected attack or infection. This was previously impossible since password hashes and tickets could remain on the local machine or even on some remote services after logout. The fact that credentials stay on a single physical device also makes quarantining an ongoing attack much easier for domain operators.

V. CAS IMPLEMENTATION AND EVALUATION

In this section a CAS implementation focusing on the *Server Message Block* (SMB) [9] protocol in version 2, with common *NT LAN Manager Security Support Provider* (NTLMSSP) [10] authentication is described. This proof of concept (PoC) has been implemented and subsequently optimized for performance to verify the general feasibility, functionality and delay-free usability of the CAS engine. It is realized in the programming language *C* with the library *libevent* [11], which provides proxy functionality based on callbacks on network socket events. As soon as the CAS proxy engine receives SMB traffic it is handed over to it's parser module. The parser extracts the RID from the NTLMSSP authentication protocol messages inside the SMB data stream. Upon identifying the client the lookup to the back end is performed to correlate the identity to a smartcard. The PoC's back end is a simple file based database that can be created with a setup tool. The PoC performance is evaluated with focus on comparing the RID parser and correlation lookup, both in user and in kernel mode. These measurements are focusing on the number of SMB/NTLMSSP authentications versus CPU workload and throughput without encryption of the protocol data channel. Thus the SMB service connection is established between client and CAS for evaluation and measurement issues separately from the 2^{nd} Factor CAS AuthN authentication protocol. CAS AuthN was realized by using standard TLS 1.2 with client certificate extension [5], [6] and is not examined further. The SMB protocol is configured to use the default plain-text mode to obtain accurate measurements without encryption overhead.

A. Exemplification & analysis based on SMB with NTLMSSP

SMB with NTLMSSP is applied widely and, besides Kerberos [12], one of the most used protocols and mechanisms in Windows based enterprise domains. Still, other affected mechanisms and services are deployed practically, so that it becomes clear that a full-featured CAS needs to understand various affected communication and authentication protocols.

To ensure that both client and server use the same protocol while exchanging data, the service dialect is negotiated at the beginning of the handshake. When the server receives the list, it responds with a chosen dialect revision, in this work SMB version 2.1. As soon as the dialect is negotiated, the client starts the authentication handshake by sending a list of supported authentication mechanisms (step 3). The server will reply with the chosen mechanism, in this work NTLMSSP, and a challenge (step 4, NTLMSSP_CHALLENGE), which has to be solved by the client. Afterwards, the client sends its response containing the solved NTLMSSP authentication challenge and data (step 5, NTLMSSP_AUTH). After verifying the response received (step 6 and 7) the server replies if the authentication was successful or not (step 8, NT_statuscodes).

Extraction of the RID: When securing the NTLMSSP authentication relevant authentication packets contain the zero-byte-delimited string "NTLMSSP", which is the marker to start offset calculation. The required data is found in the NTLMSSP structure of step 5, defined as NTLMSSP_AUTH. Note that any length and offset values in the NTLMSSP structure are not in network byte order, but in little-endian. The NTLMSSP messages contain user name, host name and domain name encoded in UTF-16LE format. They have to be converted to the CAS character encoding and a lookup in a connected credential database is started to evaluate if a user is allowed to access the requested service or not. These three identifiers

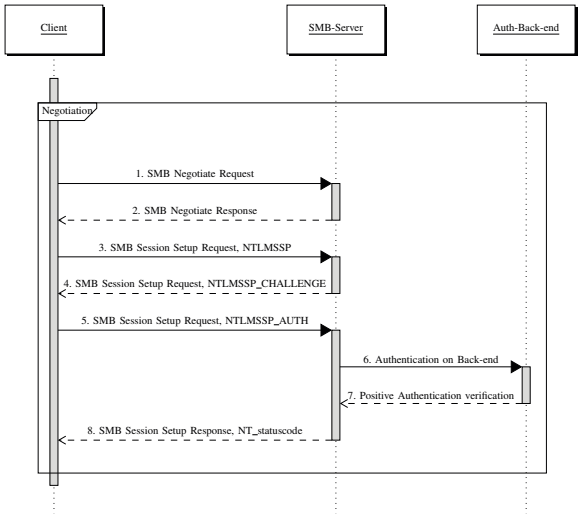

Fig. 3. NTLMSSP authentication process

form the RID. Offset positions based on the starting byte of the string "NTLMSSP" are:

- Bytes 32-35: offset for '*domain name*'
- Bytes 40-43: offset for '*user name*'
- Bytes 48-51: offset for '*host name*'

These offset positions and the corresponding *length* value, which starts 4 bytes earlier and is 2 bytes long, have to be read. The essentials of the NTLMSSP message for extracting the RID are illustrated in Figure 4.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|N T L M S S P 0| Type  |   |   |       |   |   |       |dl |   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|dl_off |ul |   |ul_off |hl |   |hl_off |   |   |       |   |   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

dl     = domain length
dl_off = domain offset
ul     = username length
ul_off = username offset
hl     = hostname length
hl_off = hostname offset
```

Fig. 4. NTLMSSP authentication message & offsets for extracting the RID

Analysing the protocol a bit more in detail shows some pitfalls when parsing the NTLMSSP structure for this information. In Windows there are two contradictory ways of formatting log-on names:

- *User Principal Names*,
  in the form: *username@domain*
- *Down-Level Log-on Names*,
  in the form: *DOMAIN\username*

It is important to note that there are even two different ways Windows handles these names in NTLMSSP messages, which requires the CAS engine to carefully distinguish these cases:

1) User Principal Names will be sent entirely in the user name field of the NTLMSSP message while the domain field contains an empty string.
2) Down-Level Log-on Names will be split correctly so that the domain field contains the domain name and the user name field contains the user name.

This inconsistent behavior allows an attacker to build malformed combinations of both formats:

DOMAIN1\username@domain2

When entering such malformed combinations, Windows fails to establish SMB sessions and falls back to the WebDAV protocol [13] that tries to establish a connection via HTTP. The NTLMSSP structure constructed in this case contains two domain names: One in the domain field, and one integrated into the user name. It is not defined and probably implementation-dependent how these malformed structures are interpreted at the server. Thus the obvious countermeasure is to detect such malformed combinations and block them at the CAS. Moreover, there are further fall-backs like NetBIOS [14]. It is important to deactivate all possible fall-backs by the CAS, so that only implemented and known protocols are processed while unknown mechanisms have to be blocked by default (*whitelist design*). Another NTLMSSP issue is that the protocol uses offset and length fields, and thus pointers into the authentication message, which are prone to buffer overflow attacks. The CAS must thus verify the pointers and make sure they do not point outside the received message while parsing the RID. To all intents and purposes, CAS has to parse all fields of all messages accurately so that other attack vectors like segmentation attacks, buffer overflows etc. are considered and disabled by design.

B. Performance evaluation - User vs. kernel mode

The PoC redirects incoming connection attempts to its sockets, it thus works as a transparent proxy. In the second version of the CAS this user space proxy was ported to kernel mode. The CAS engine was expected to gain a significant performance improvement if the network I/O operations would be handled in kernel space, so that the expensive context switching, which is needed when transferring the network data to user space, can be omitted completely when operating on the incoming TCP streams directly. In this case only the identity management and correlation is handled in user space. The results are presented in the following by comparing the user space implementation with the kernel mode version.

A *Stormshield SN500 appliance* [15] is connected in between two gigabit network ports of a *Spirent Avalanche* [16] Packet Generator as a test-setup. SN500 is based on FreeBSD version 9.2 and runs the introduced CAS PoC and configuration as described before. SN500 is equipped with a *VIA Nano U3100 1.3Ghz* and provides *1 GB RAM*. The benchmark is done by finding the point where there are close to no unused CPU cycles left (99-100% work load) to handle new connections. The CPU is then running at maximum workload for 2 minutes. The following benchmarks are averaged from the results.

Auth per second: The first benchmark is based on minimal SMB/NTLMSSP sequences. After the authentication request has been sent, the RID is being identified and after the required identity lookup has been carried out, the session will be closed immediately. This benchmark illustrates the worst case where all connections are trying a bad password in a loop. This simulates a distributed login brute force attack.

Throughput & latency: The second benchmark simulates a normal user session with some data transfer. The measurement contains a successful authentication followed by a download of a 1 MB file. Latency is measured during the benchmark.

CAS in proxy mode: The kernel that runs on the SN500 has a Network Address Translation (NAT) rule set to divert the traffic to the CAS. It receives these connections on the loopback interface, a lookup into the NAT table is done to retrieve the original destination. The CAS is responsible to decode, analyze and copy all network data between the sockets in user space. Moreover, a performance evaluation with *Callgrind* [17] was performed, which showed that most of the CPU cost resides in sending, receiving and forwarding the TCP segments on the libevent buffers. Only 2.2% of the time is spent in the implemented code of the CAS parser, thus performance improvements were suspected when moving buffer management to kernel space.

CAS in kernel mode: In this PoC version, the code to decode and analyze the SMB traffic is running in kernel space, being able to handle network data without context switching. As soon as the NTLMSSP authentication is detected and the user is identified, the connection is paused and the kernel engine simply asks the CAS user space engine to fulfill the correlation lookup. The result is returned to the kernel and on success the paused connection is continued by the kernel engine.

Table I shows the comparison between the reference measurement where the CAS was not enabled, the user space proxy, and the kernel mode PoC running on the SN500.

	Reference	Proxy	Kernel
Auth./sec	2300	500	2200
Throughput (Mbit/s)	670	250	670
Latency (ms)	<1	5	<1

TABLE I
COMPARISON OF THE CAS ENGINE IN USER AND KERNEL MODE

Summarizing, Table I emphasizes the benefits when porting the approach from user to kernel space resulting in much better performance of the CAS approach with regards to possible authentication amount, throughput and latency. Therefore, parsing proves to be very efficient as only certain SMB/NTLMSSP header fields need to be evaluated. Moreover, the CAS PoC uses a hash table for the identity lookups in order to correlate the NTLMSSP identity with the TLS connection identity. The hash table provides add/find/delete operations in constant time complexity O(1). Thus no additional penalty is expected when the number of concurrent users is increasing.

VI. CONCLUSION AND FUTURE ALIGNMENTS

This work addresses the Pass-the-Hash problem at its root and offers the proper authentication mechanism, which is missing in MS domain networks. The important functionality of the CAS is access management and the integrated CAS AuthN, which protects the insecure SSO service authentication SRV AuthN with verifiable token authentication, e.g. by smartcard.

In the previously introduced scenarios the user simply enters the PIN when logging onto the SSO system. No further steps are needed if the client-side CAS extension is started by the initial smartcard SSO login. Moreover, an attacker has no possibility to execute a PtH attack without matching CAS AuthN credentials, that is without the correct user's smartcard. The described identity correlation also results in a strong protection mechanism against insider attacks. In conclusion, inside and outside attackers would have to steal or duplicate each single smartcard including all PINs of each target user to gain access to protected services with a foreign identity. Thus lateral movement is consequently impossible.

Furthermore, it becomes clear that the CAS has to process requests with *whitelist principle*, so that all known and unknown types of fall-backs, dialects of transport protocols, (yet) unsupported authentication mechanisms and remote services are blocked by the CAS engine by default.

With other affected services in mind, which are offering an additionally encrypted version of their protocol, the RID extraction from the protocol's session is complicated. Here, a CAS needs to inspect the encrypted traffic so that further key deployment may be required, that enables intercepting a connection on-the-fly with the help of a trusted Certification Authority (CA). Another promising strategy would be to simply disable any Microsoft encryption as a default policy, because encryption keys for their mechanisms are derived from the insecure password hashes the attacker already possesses. The connections are already secured by the the smartcard channels of CAS, which encrypts the connection by the 2^{nd} Factor token. Handling encryption as supported by SMB version 3, or other SSL/TLS protected service protocols, are a focus for further research.

To establish a flexible solution in practice, most of the commonly used and affected Microsoft protocols must be supported by the CAS and they will have to be identified and evaluated in the next steps. Besides SMB this includes HTTP, Exchange (DCE/RPC), IMAP, POP, SMTP etc., which can also be configured to use both NTLMSSP and the Kerberos authentication [12]. As sometimes falsely assumed, Kerberos offers no protection against the PtH problem [18], [19]. In general, Kerberos just replaces the password hashes with tickets. There are two kinds of tickets, the domain's Key Distribution Center (KDC) issues the *Ticket Granting Ticket* (TGT) upon successful user login. The TGT allows users in subsequent requests to obtain *Session Tickets* (ST) for service sessions from the KDC. Similar to a PtH attack, these tickets can be copied and authenticate the user as the NTLM hash does. The shown CAS approach can be aligned to support the Kerberos mechanism. A brief analysis of the Kerberos protocol tracking the tickets without decrypting them shows a promising approach in correlating the ticket to the legitimate owner. In this case the CAS needs to execute the following steps:

1. CAS identifies the client on SSO login and extracts the identity from the request of the initial TGT issuance.
2. Afterwards CAS tracks when the client requests an ST by a known TGT, to which the RID has already been

correlated in step 1. The RID is thus handed over from the TGT to the ST, issued by the KDC for further service access.

3. CAS is now in possession of the RID correlated to both TGT and ST, so that the client can be forced to show its CAS AuthN token before access with a ST will be granted.

Further studies will identify the best solution that will be able to handle Kerberos authentication controlled by the CAS engine.

Besides the already mentioned benefits with regards to security, simple integration and usability, the existing and insecure service protocols and authentication mechanisms can remain untouched and do not have to be modified. Moreover, other vulnerable protocols besides those affected by PtH can be secured by a CAS implementation, which is able to authenticate any user if it is possible to extract any RID by related service requests. As far as a RID can be defined to be correlatable to a unique requesting machine's characteristic, the approach can be aligned for different use cases to improve different authentication mechanisms. Further studies have to be executed to generalize this solution in order to protect against common password theft and identity misuse in high security areas of enterprise networks and SSO domains.

REFERENCES

[1] ISMG, Palo Alto Networks, "Adavanced Persitent Threats Survey New Strategies to Detect, Prevent and Defend," 2014. [Online]. Available: https://www.ismgcorp.com,https://www.paloaltonetworks.com/

[2] C. Hummel and J. Niem, *Why Crack When You Can Pass the Hash?*, SANS Institute, 2009, http://www.sans.org/reading-room/whitepapers/testing/crack-pass-hash-33219.

[3] A. Duckwall and C. Campbell, "Still Passing the Hash 15 Years Later? Using the Keys to the Kingdom to Access All your Data," *Blackhat*, 2012, briefing and Keynotes, https://www.blackhat.com/html/bh-us-12/bh-us-12-briefings.html, https://media.blackhat.com/bh-us-12/Briefings/Duckwall/BH_US_12_Duckwall_Campbell_Still_Passing_Slides.pdf.

[4] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301 (Proposed Standard), Internet Engineering Task Force, December 2005, updated by RFC 6040. [Online]. Available: http://www.ietf.org/rfc/rfc4301.txt

[5] N. W. Group, "RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2," 2008.

[6] I. E. T. Force, "RFC 5878 Transport Layer Security (TLS) Authorization Extensions," 2010. [Online]. Available: https://tools.ietf.org/html/rfc5878

[7] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, 1978.

[8] A. Oberle, P. Larbig, N. Kuntze, and C. Rudolph, "Integrity Based Relationships and Trustworthy Communication between Network Participants," in *IEEE ICC 2014 - Communication and Information Systems Security Symposium, (ICC'14 CISS)*, Sydney, Australia, June 2014, pp. 610–615.

[9] *Server Message Block (SMB) Protocol*, Microsoft Corporation, May 2014. [Online]. Available: https://msdn.microsoft.com/en-us/library/cc246231.aspx

[10] *NT LAN Manager (NTLM) Authentication Protocol Specification*, Microsoft Corporation, Oct 2008. [Online]. Available: https://msdn.microsoft.com/en-us/library/cc207842.aspx

[11] N. Mathewson and N. Provos, "libevent an event notification library." [Online]. Available: http://libevent.org/

[12] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, "The Kerberos Network Authentication Service (V5)," RFC 4120 (Proposed Standard), Internet Engineering Task Force, July 2005, updated by RFCs 4537, 5021. [Online]. Available: http://www.ietf.org/rfc/rfc4120.txt

[13] L. Dusseault, "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)," RFC 4918 (Proposed Standard), Internet Engineering Task Force, June 2007, updated by RFC 5689. [Online]. Available: http://www.ietf.org/rfc/rfc4918.txt

[14] N. W. G. in the Defense Advanced Research Projects Agency, I. A. Board, and E. to End Services Task Force, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods," RFC 1001 (Standard), Internet Engineering Task Force, March 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1001.txt

[15] Arkoon and Netasq, subsidiary of Airbus Defence and Space, "Stormshield SN Series - Unified Security." [Online]. Available: www.stormshield.eu

[16] Spirent, "Avalanche - Application Performance/Security Test, Extreme Scale and Performance." [Online]. Available: http://www.spirent.com/

[17] Valgrind Authors, "Callgrind: a call-graph generating cache and branch prediction profiler." [Online]. Available: http://valgrind.org/docs/manual/dist.authors.html,http://valgrind.org/docs/manual/cl-manual.html

[18] A. Duckwall and B. Delpy, "Abusing Microsoft Kerberos - Sorry you guys don't get it," *Blackhat*, 2014, briefing and Keynotes, http://de.slideshare.net/gentilkiwi/abusing-microsoft-kerberos-sorry-you-guys-dont-get-it.

[19] CERT-EU, *The Kerberos Golden ticket*, Computer Emergency Response Team, July 2014, http://www.sans.org/reading-room/whitepapers/testing/crack-pass-hash-33219.