USENIX Security '24 Artifact Appendix: Windows into the Past: Exploiting Legacy Crypto in Modern OS's Kerberos Implementation

Michal Shagam Eyal Ronen

Tel-Aviv University

A Artifact Appendix

A.1 Abstract

The Kerberos protocol is used by millions of users and network administrators worldwide for secure authentication, key distribution, and access control management to enterprise networks and services. Since its initial public deployment in 1989, the protocol has undergone many revisions to incorporate new cryptographic primitives and improve security. For example, initially based solely on users' passwords and symmetric cryptographic primitives, current implementations also support smartcard-based authentication with asymmetric cryptographic primitives for improved security. However, this iterative revision process has resulted in implementations riddled with legacy crypto primitives and protocol designs.

In this work, we show how we can exploit this legacy crypto to completely break the security of the enterprise network. Firstly, while arguably more secure, smartcard-based authentication uses RSA encryption with the notorious PKCS #1 v1.5 padding scheme. Although the RSA decryption is done securely inside the smartcard, a non-constant time unpadding code runs on the client's CPU. This makes both Windows's and several Linux distributions' implementations vulnerable to the Bleichenbacher attack that can recover cryptographic session tokens. Secondly, we show that the RSA smartcard-based authentication does not provide forward secrecy to the cryptographic tokens that the server provisions to the client. Thirdly, we propose and analyze different algorithmic approaches to minimize the overhead required to handle noisy oracles in the Bleichenbacher attack. This general Bleichenbacher attack analysis may be of independent interest.

Finally, we demonstrate microarchitectural side channel-based end-to-end attacks on the Windows Kerberos implementation. We start by showing how to recover tokens used to encrypt session transferred remote files by Samba. We then show how to amplify the number of decryptions performed with a single user's PIN code input, allowing us to accelerate our attack and recover users' (and admins') credentials before expiration. In addition, we describe a remote attack vector that allows us to perform the attack and generate queries.

We provide attack evaluation artifacts offering access to a network setup of our attack model, code implementing both the client side native attack code and the malicious machine-in-the-middle attacker code for our end-to-end-attack, a detailed tutorial on how to use the attack code and tools provided, code for classification and detection of messages, data from experiments performed and code generating the graphs shown in the paper. The code we provide for packet modification can additionally be utilized as a tool enhancing network security analysis.

A.2 Description & Requirements

We will provide access to a live setup including instructions on how to access it and create a similar setup. The artifacts which will describe the major claims and experiments to be evaluated will be available in a dedicated repository (Section A.2.2) along with a tutorial containing instructions and steps for attack benchmarks.

A.2.1 Security, privacy, and ethical concerns

We are providing a live setup and do not expect to modify anything on the evaluators' end. We started a responsible disclosure process with Microsoft, the maintainers of the Heimdal Kerberos implementation, and the OpenSC maintainers. OpenSC have fully patched their PKCS #1 v1.5 code to be constant-time. We were allocated CVE-BLINDED with a high severity jointly with a disclosure from a concurrent work. Microsoft acknowledged our findings as an Elevation of Privilege attack. They are currently working on a fix which they plan to release in July 2024. Although they don't have access to exact numbers, they estimate that a large number of users use smartcards for Kerberos authentication and are affected by our findings. The Heimdal team has confirmed our findings, and we are waiting for updates regarding their patching plans.

A.2.2 How to access

The artifacts will be available in a dedicated Github repository: KerberosSmartcardPaddingOracleAttack
The repository will include:

- Attack Documentation
- Attack threat model description
- Attack source code
- Simulation source code
- Data from experiments performed
- Graph generation code
- Installation instructions
- Tutorial and instructions for accessing the live setup

We will provide access to the Client and Server components. For simplicity, on the live setup we will run an equivalent MiTM on the Server component that will modify the packets before they reach the Server.

A.2.3 Hardware dependencies

Our threat model assumes a network with honest servers and an honest user trying to log in using an uncompromised smartcard and client machine. A malicious MiTM is able to intercept and modify packets over the network and to communicate with an unprivileged malicious program running on the Client.

Therefore we will address the requirements for the different components in the system and refer to them as the Smartcard component, Client component, MiTM component, and Server component.

Smartcard Component Smartcard that can perfrom RSA decryption. The current setup uses a YubiKey 5 NFC CSPN smartcard which can support RSA keys up to 2048 bits.

Client Component The requirements for the microarchitectural attack hold (Flush+Reload). The current setup has an Intel dual core CPU (i7-7600U).

MiTM Component A component on the network that can hijack and modify network packets before they reach the Server component.

Server Component The requirements for Windows Server hardware requirements hold.

A.2.4 Software dependencies

Our end-to-end attack targets a Client running a Microsoft Windows operating system that allow the use of RSA smartcard based login.

For our live setup:

Client Component Windows 10 Pro Version 22H2 OS build 19045.2486.

Server Component Windows Server 2019

A.2.5 Benchmarks

None

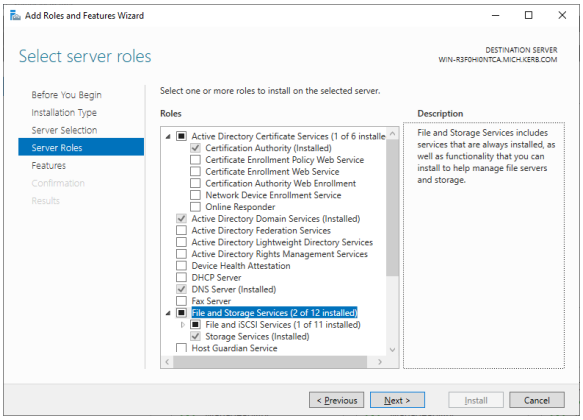

Figure 1: Role based Active Directory Server Configuration

A.3 Set-up

In addition to the provided live setup, we will provide instructions for reproducing the setup.

Live setup information:

Windows build 19045 (Verified up to July 2024 - artifact creation)

Client setup used a Dell Inspiron 5520 with Dual core intel KabyLake.

Server version Windows Server 2019 (Verified up to July 2024 - artifact creation)

Server is a virtual machine set at 2 cores and hosted on a NUC.

A.3.1 Installation

Server Component This component can be installed on a virtual machine. Install Windows Server 2019. Install Active Directory as a role-based installation and install the roles as shown in Figure 1.

Use the Yubikey tutorial to create a certificate template and choose the default options which are RSA and 2048 bits: Setting up Smart Card Login for Enroll on Behalf of

Generate user credentials for the smartcard: **Smartcard Component** Smartcard deployment

Client Component Install the required drivers for the smartcard used. After the Server Component is running add the Client to the domain. Server host may need to be added to the hosts file.

MiTM Component In order to allow hijacking and modification of the network packets, the WinDivert tool and pydivert should be installed on either the Server or on any component in the network flow between the Server and the Client.

A.3.2 Basic Test

Verify Smartcard Login Verify that the smartcard login is being performed correctly. Run network tracing tool on the

Client and perform a Smartcard login. Apply packet filter "kerberos.msg_type ==11". If no packets are found, the smartcard network login isn't being performed. This can happen when the Client has a network issue and uses the cached credentials to perform the local login.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Our End-to-End attack can decipher tokens using the padding oracle. This is proven in (E1) where the end-to-end attack will be used to decipher a token.

(C2): Our detection with early abort attack can find messages likely to be deciphered in less than 10 hours. This is shown in (E2).

A.4.2 Experiments

(E1): End-to-End attack on a fast message [20 human-minutes + 10 compute-hour + 16 GB RAM]: The attack will be performed on a message that can be deciphered by a perfect oracle in under 10k queries.
How to: On a setup as described in the threat model in Section 3.2, run the End-to-End attack with the example message.
Preparation: Connect a smartcard to the Client. Run native malicious code on the Client computer from any user (preferably not the victim). Run MiTM attacker in End-to-End attack mode and verify it has connected to the malicious code running on the Client.
Execution: Perform actions as the victim generating queries. This can be done either by performing actions such as trying to open a remote file or by opening the example local html file in a browser such as Google Chrome or Microsoft Edge.
Results: Results can be found on the MiTM attacker which will output information on how long the experiment took and the deciphered message as well as information on the amount of false negatives or false positives and how the traceback method was used.
Notes: The attack can be performed on additional messages and will output information on the amount of unknown bytes. In cases where one of the hosts has gone to sleep, we had to restart the browser to keep generating queries. We will be glad to assist to help everything run smoothly.

(E2): Detect and Early Abort attack [60 human-minutes + 50 compute-hour + 16 GB RAM]: The attack will attempt to find a fast message that can be deciphered in less than 10 hours for different messages generated by the server.
How to: On a setup as described in the threat model in Section 3.2, run the Detect and Early Abort attack.
Preparation: Connect a smartcard to the Client. Run native malicious code on the Client computer from any user (preferably not the victim). Run MiTM attacker in Detect and Early Abort attack mode and verify it has connected to the malicious code running on the Client.
Execution: Perform actions as the victim generating queries. This can be done either by performing actions such as trying to open a remote file or by opening the example local html file in a browser such as Google Chrome or Microsoft Edge.
Results: Results can be found on the MiTM attacker which will output information on the amount of messages attempted, how many fast messages were found and how long the attack took. For deciphered messages, the deciphered message will be given as well as information on the amount of false negatives or false positives and how the traceback method was used.
Notes: The attack can be performed with different limits for the amount of queries. In cases where one of the hosts has gone to sleep, we had to restart the browser to keep generating queries. We will be glad to assist to help everything run smoothly.

A.5 Notes on Reusability

On a different domain, the public key for the victim needs to be added to the code. The attack doesn't rely on the use of a specific Smartcard.

While the end-to-end attack targeted a Windows Client, several adaptations can be made to allow additional setups. Although we didn't check, we have no reason to believe the query generation methods our attack used on Windows can't be extended to other operating systems and configurations with an equivalent oracle.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.