Attacking and fixing the Microsoft Windows Kerberos login service

Malgherini Tommaso

UNIVERSITÀ CA' FOSCARI - VENEZIA

Dipartimento di Informatica

Kerberos – what it is (in short)

- Protocol for authentication inside an insecure network
- Client-server model, with a “trusted third party” for credentials distribution
- Based on **simmetric cryptography** and the **Needham-Schroeder** protocol

Most known implementations

- **Kerberos MIT:**
The original implementation developed at MIT, released under BSD license.

- **Windows Domain Controller:**
From Windows 2000 onwards Microsoft adopted Kerberos 5 as their default network authentication protocol. Initially only **DES** and **RC4** were supported, from Windows Vista **AES** is used by default

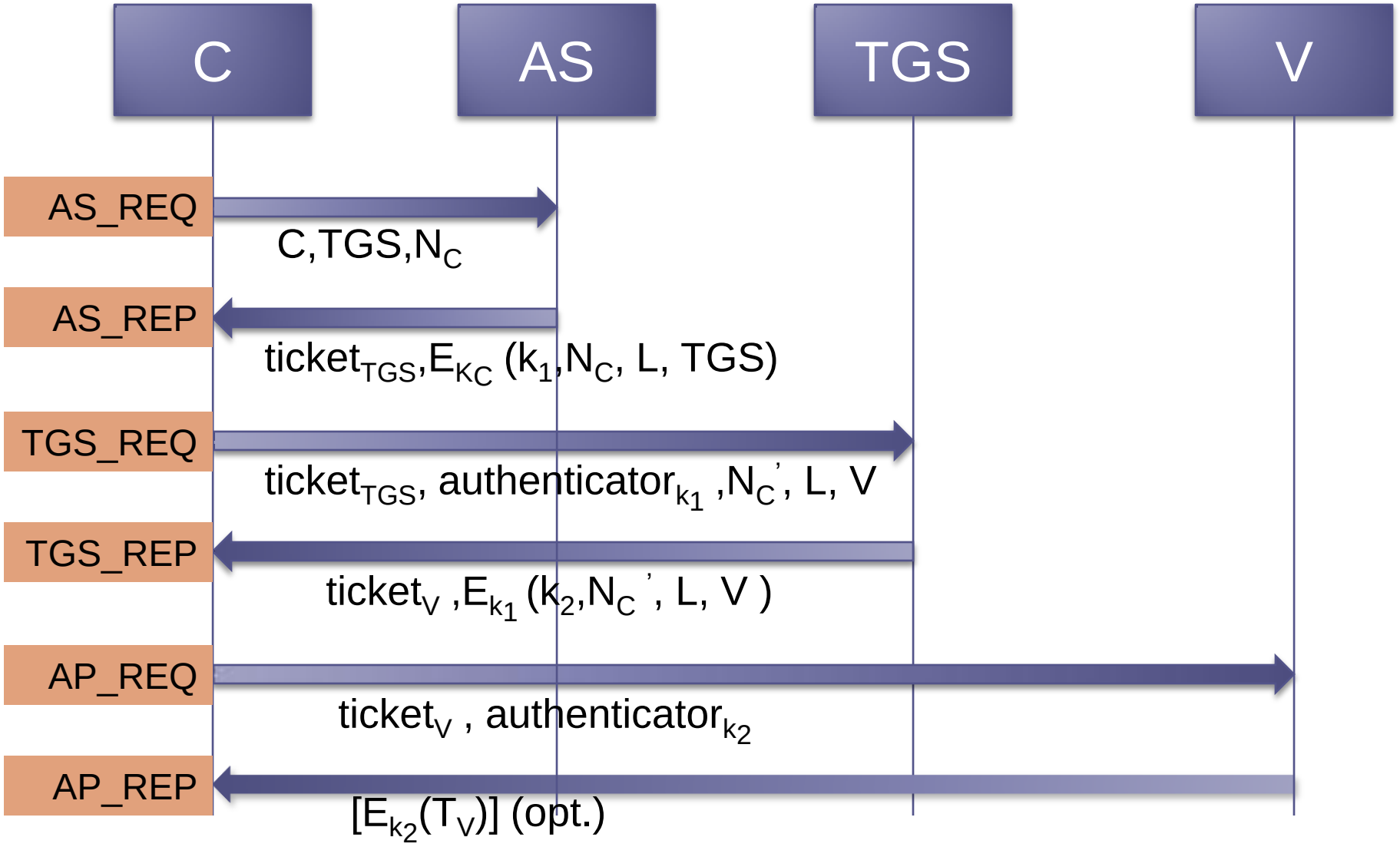
- **Heimdal:**
An alternative to the MIT implementation, developed in Sweden .

Notations

- **C:** Client
- **AS:** Authentication Service
- **TGS:** Ticket Granting Service
- **V:** Verifier (== service server)
- **L:** Lifetime
- **T_A :** Timestamp from A's clock
- **K_A**: Personal key of the entity A (for the clients password => key)
- **k_n:** Session key
- **N_A:** *Nonce*
- **$E_K(x)$:** A message *x* encrypted with key *K*

- **$ticket_V$** = $E_{K_V}(k_n, C, L)$
- **$authenticator_{k_n}$** = $E_{k_n}(T_A, [k_m])$

Kerberos 5:How it works



Kerberos 5: Why it's a good thing

- **Single Sign-On**, after the initial request to the TGS service, the ticket is cached and reused for each service request (no need to use a password anymore)

- Only the first request is encrypted with K_c, temporary session keys are used for every following request.

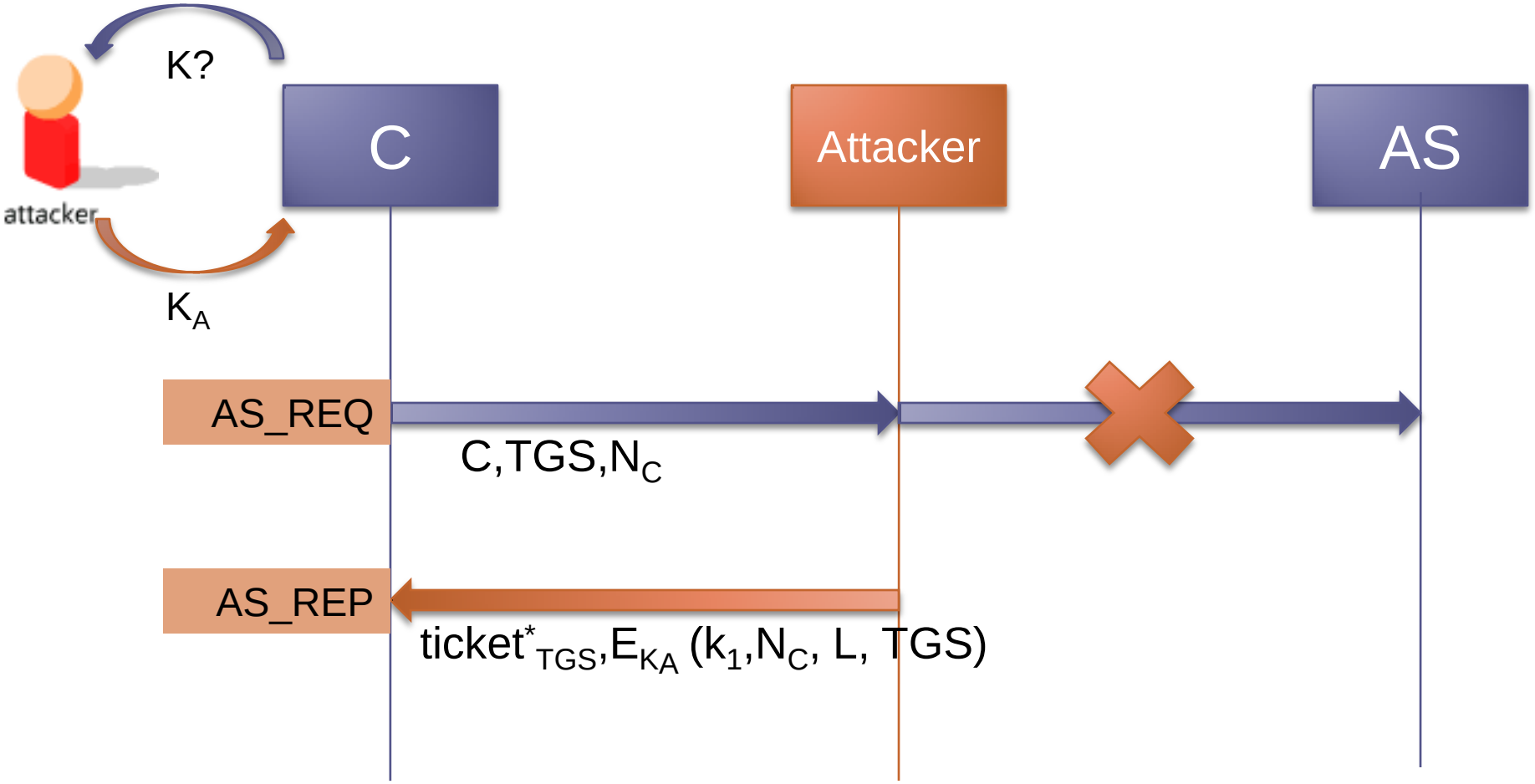
Known attacks

1. KDC Spoofing

- We'd like to use kerberos with the physical login process on the network workstations.
- This happens by default on Windows domains since Windows 2000 and can be done on Unix/Linux with a specific PAM module, pam_krb5
- But the LOGIN operation is not a service like the others
- The third exchange (AP_REQ/AP_REP) doesn't seem to have any sense here.
- Initially, the PAM module implemented only the first exchange
- The user inserted the password from which a key was derived. Then an AS_REQ request was made, and then the key was used to decrypt the AS_REP reply.
- If it decrypts correctly, it must be that the user knows the password....right?

Known attacks:

1. KDC Spoofing



Known attacks:

1. KDC Spoofing - Mitigation

- Publicly releasead in 2001 by Dug Song
- Solution adopted: insert also TGS_REQ/REP exchange in the login process

- For every workstation in the network a principal like: host/<nomehost>@<realm> is created
- Secret keys for each of these services are exported in the respective machines

- After AS_REP is verified, the client machine asks for a ticket for the login service

- Since only the KDC and the client machine know the key, if the produced ticket decrypts correctly, the login is permitted

Known attacks:

1. KDC Spoofing - Sidenote

•A final note about this attack:

•On *nix systems, secret keys are exported inside a “keytab”, a file with root-only access (for obvious reasons)

•If the pam_krb5 module is called from a non-root process, **the ticket cannot be verified.**

•PAM will fail back to using only AS_REQ/AS_REP, leaving the system vulnerable

• example of typical *nix process using PAM without root privilege: every kind of screensaver lock

Known attacks:

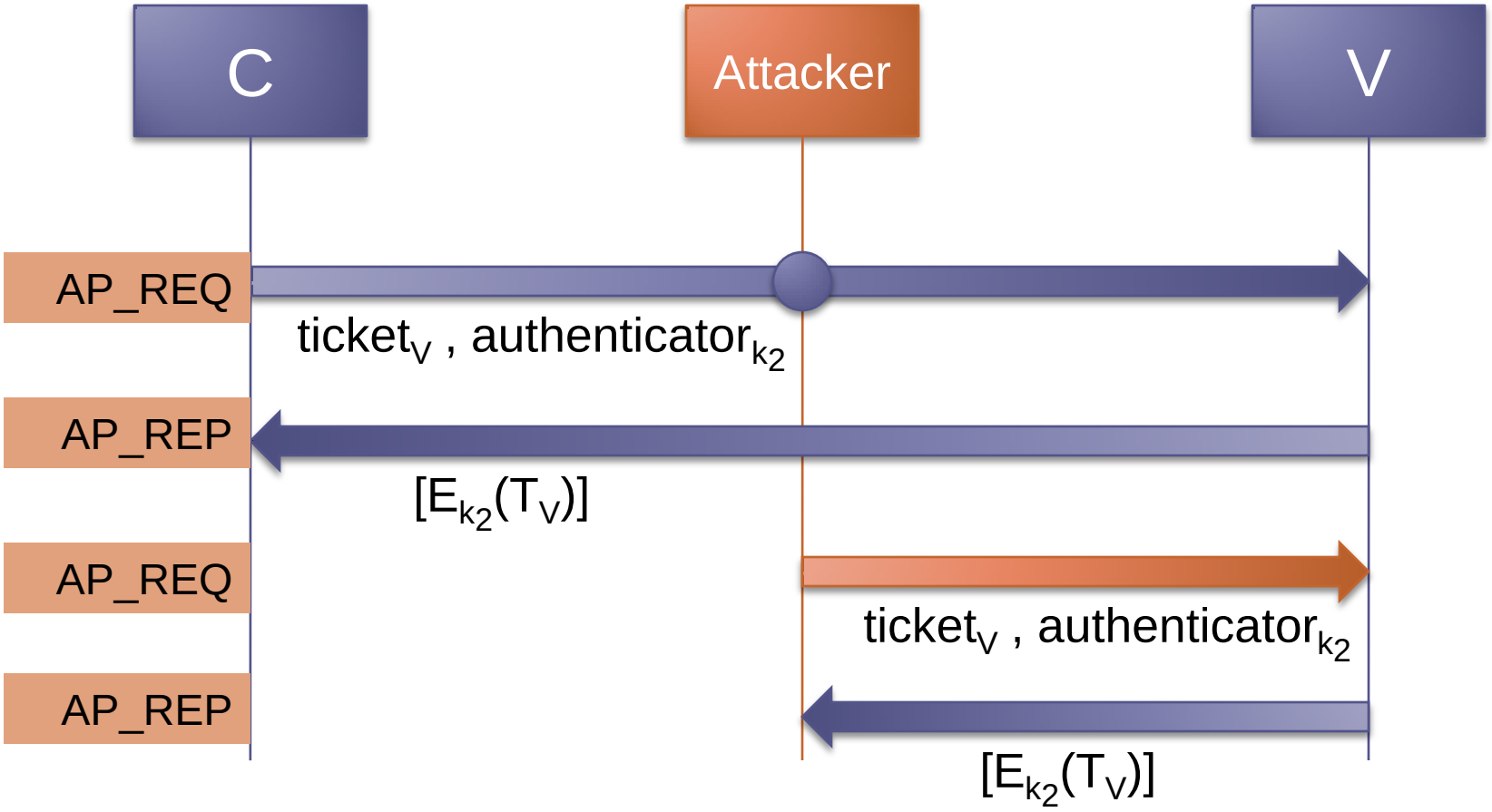
2. Replay Attack

- It can be seen that in the message flow, the critical one is AP_REQ, since that alone in the end decides if the user gets authenticated

- Is it possible to reuse an old AP_REQ?
- **Timestamp** in the authenticator: old requests are dropped
- But Kerberos allows for a certain time window (5 min.) from the timestamp, inside which an authenticator can be accepted
- The lifetime for the ticket is much more longer, usually 24h.

Known attacks:

2. Replay Attack



Known attacks:

2. Replay Attack – Mitigation

•Known for a looong time (discussed in ‘91 USENIX Proceedings, for Kerberos IV).
•Both RFC 4120 and its predecessor, RFC 1510 specify that an already accepted authenticator MUST be rejected.
•Authenticator are cached for a certain time and maintained at least until the time window for its acceptance is expired
•So it’s just another “boring” theoretical attack?

•And if the attacker can actively delete legitimate requests and then uses the stolen authenticator for himself?
•This variant was demonstrated to be successful against the SMB service on a Windows 2000 SP 3 server in 2003 by Kasslin and Tikkanen.

The “pass-the-ticket” attack

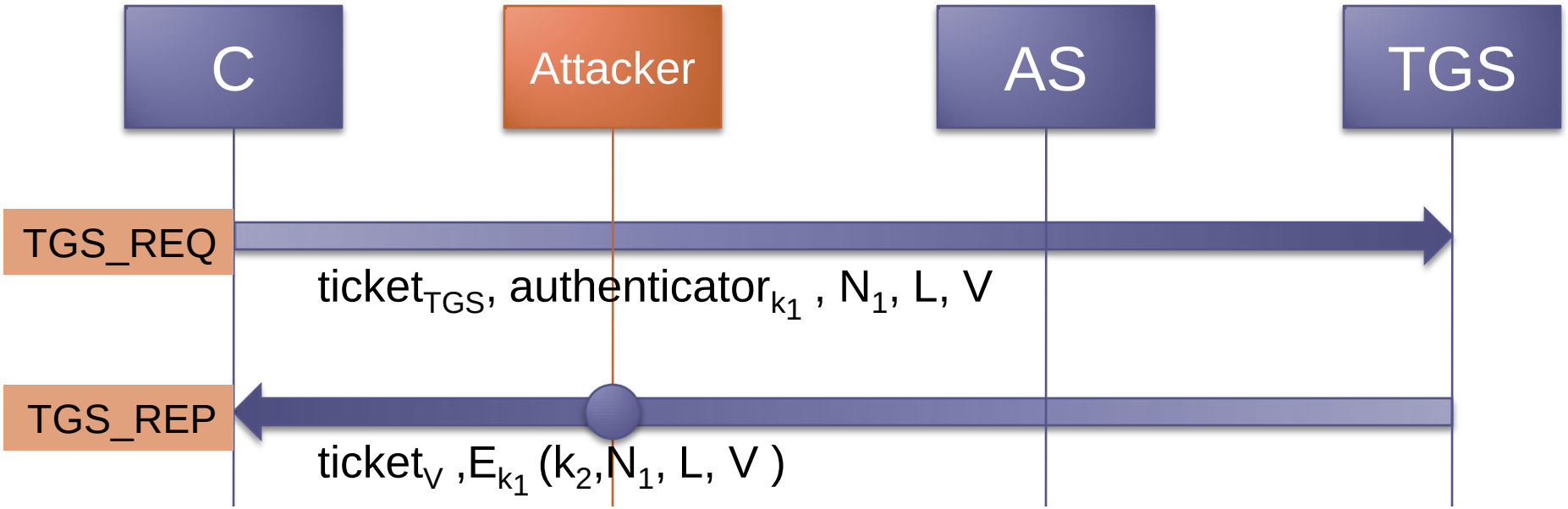
- We’ve seen that there is no AP_REQ/AP_REP exchange for the login service.
- Last message is TGS_REP...

- Could we reuse the ticket inside and old TGS_REP?

- **IDEA**: Spoofing the initial AS_REP + replay old ticket to complete the second step

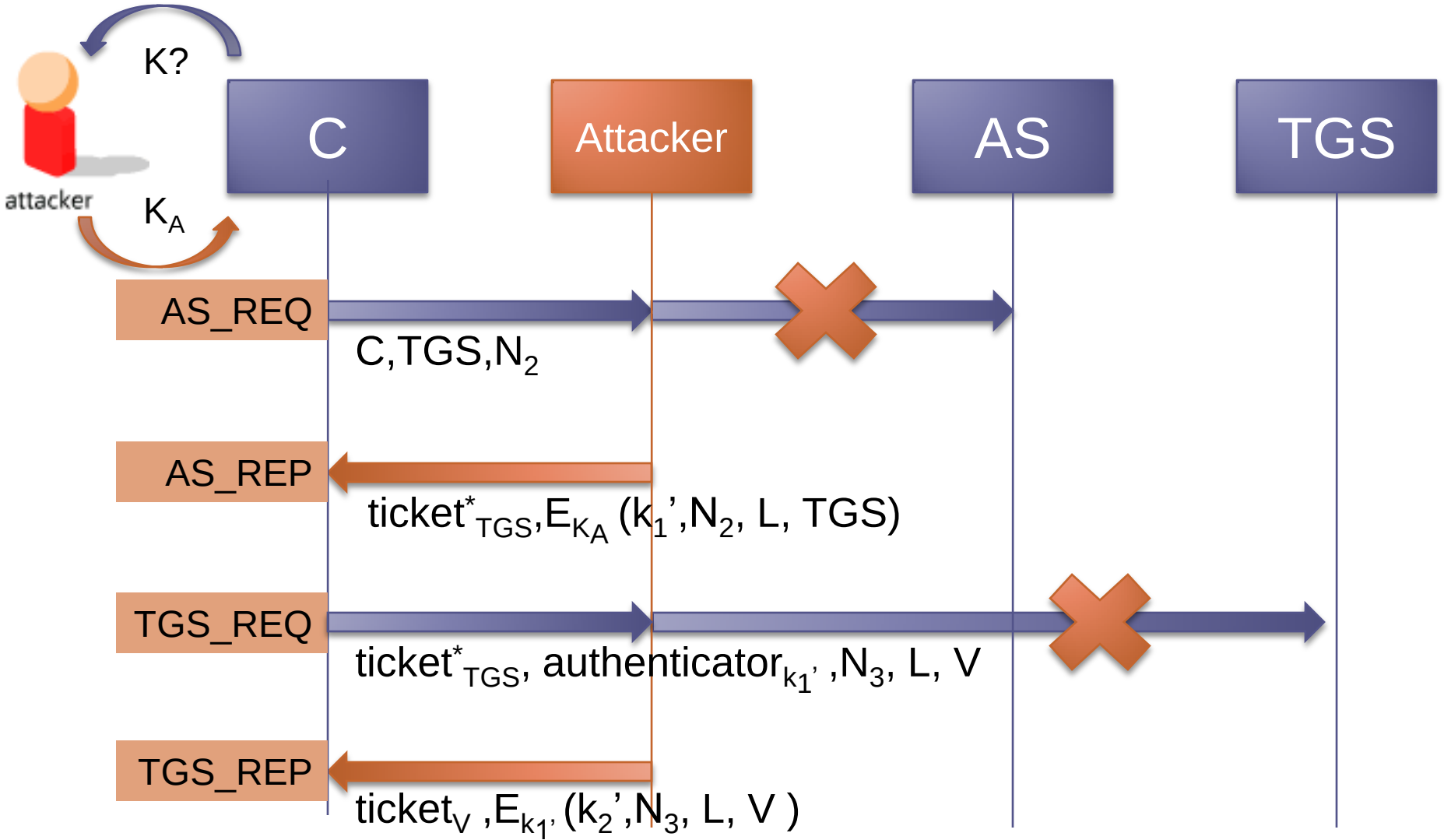
The “pass-the-ticket” attack

Phase 1:



The “pass-the-ticket” attack

Phase 2:



...and such a thing can really work??

The attack was tested against the following platforms:

- **Linux**:
  - KDC: GNU/Linux Debian 4.0, Kerberos MIT 1.6.3
  - Client Workstation:
    - GNU/Linux Gentoo, Kerberos MIT 1.6.3-r6
- **Windows**:
  - KDC:
    - Windows Server 2008
  - Client Workstation:
    - Windows XP Service Pack 2
    - Windows Vista Enterprise Service Pack 1
    - Windows 7 Professional

Results:

- Kerberos MIT is not affected by the attack

- All Windows systems were found vulnerable(!!)
- The problem is that only the ticket is checked, we have no way to tell if the sender actually knows the key

- Why the MIT implementation is not vulnerable?
- Let's analyze what's happen inside their Kerberos and see how not only it resolves our problem, but also shows us how the login process can be implemented to make it much more compliant to the protocol

Kerberos MIT

•Instead of just verifying the ticket validity by decrypting it, the received credentials are used to **internally simulate the last exchange AP_REQ/AP_REP**

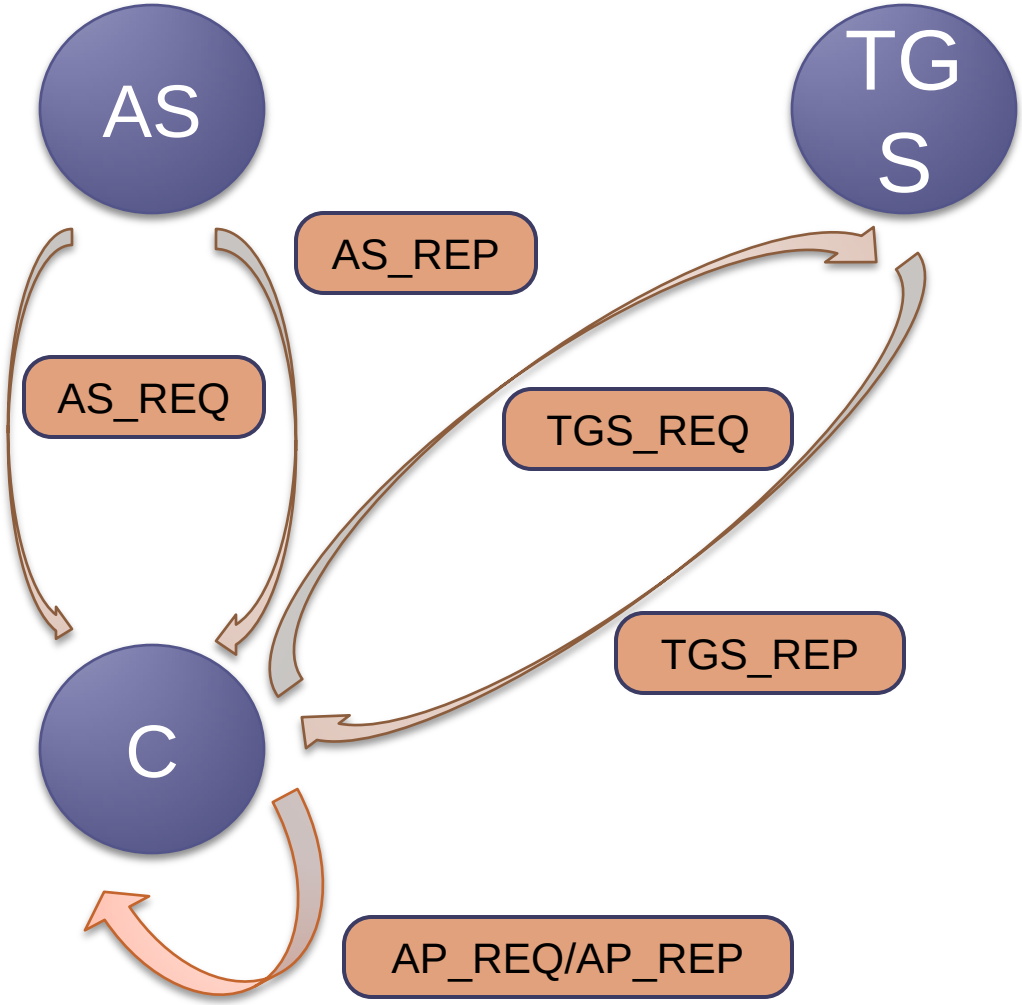
•An AP_REQ request is created using the session key contained in the non-ticket part of the TGS_REP message.

•Then, impersonating the verifier, the AP_REQ is decrypted using the key found inside the received ticket.

•The attacker just replayed that ticket and has no way of knowing the contained session key. This means that a different k_n will be used for encryption and decryption, causing a decryption error and consequent failure of the login process

•**GAME OVER**

Kerberos MIT



TGS_REP:
$E_{k_1'}(k_2', N_3, L, V), E_{K_V}(k_2, C, L)$

T_V

AP_REQ: $\text{Authenticator}_{k_2'}, T_V$

Extract_key(T_V) = k_2

Verify_ap_req():
decrypt(AP_REQ, k_2)

ERROR!!

Conclusions

- Kerberos surely provides authentication in a very secure and scalable way.
- BUT! Very much attention must be paid to the actual implementation, especially when dealing with particular cases or situations for which Kerberos wasn't designed for.
- We've seen a perfect example of this, an attack which targets a very specific service which allows for complete authentication bypass, due to an oversimplification of the implementation.
- The vulnerability was notified to CERT and Microsoft in February, who acknowledged the problem and will patch it with the next service pack.
- Proof-of-concept code was published in August and can be obtained here: http://secgroup.ext.dsi.unive.it/kerberos/

Kdcreplay.py

```
"Pass the Ticket" attack
 (a m0t Studios production)
Usage:
./kdcreplay.py [opts]

Options:

    -t <target>         set target machine ip (can be overriden in code)
    -k <kdc>            set kdc ip (same as above)
    -u <user>           set principal name (aka krb user, default="test")
    -p <passw>          set principal password (will be used as principal
secret key, default="password")
    -P <pcapfile>       skip service ticket sniffing and load dumped TGS_REP
from file
    -d <dumpfile>       save sniffed TGS_REP to file
    -r <realm>          set realm name
    -S                  skip tickets replay (kdc spoofing attack)
    -e <3des|rc4win|aes>        set encryption type (for rc4win binary key
has to be set, see code, default:3des )
    -l                  set max number of as_rep to send
    -D                  lots of debug printing
    -s                  skip spoofing and replay (for debug purposes)
    -h                  read this
```

Bibliography and links

1. Kerberos: The Network Authentication Protocol, http://web.mit.edu/Kerberos/
2. D. Song. http://www.monkey.org/~dugsong/kdcspoof.tar.gz
3. C. Neuman, T. Yu, S. Hartman, e K. Raeburn, RFC 4120: The Kerberos Network Authentication Service (V5), 2005
4. K. Jaganathan, L. Zhu, e J. Brezak, RFC 4757: The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows, 2006.
5. E. Bouillon, Gaining access through kerberos, in PacSec, 2008.
6. E. Bouillon, Taming the beast: Assess kerberos-protected networks, in Black Hat EU, 2009.
7. S. M. Bellovin e M. Merritt, Limitations of the kerberos authentication system, in Usenix Proceedings, Dallas, TX, 1991.
8. Linux-PAM. http://www.kernel.org/pub/linux/libs/pam/.
9. Scapy. http://www.secdev.org/projects/scapy/
10. CERT Coordination Center, Carnegie Mellon University, www.cert.org
11. K Kasslin and A. Tikkanen, Replay attack on Kerberos V and SMB, http://www.tkk.fi/~autikkan/kerberos/docs/phase1/pdf/LATEST_replay_attack.pdf

Questions?