Cryptographic RBAC Compiler

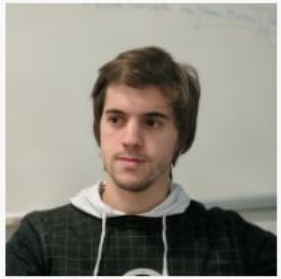
FONDAZIONE BRUNO KESSLER

Layout of the slides by Coman Catalin Andrei

Agenda

- Who am I
- The Topic
- My Contribution
- Questions

Who am I



BERLATO STEFANO

Computer Scientist

Computer Science Master Degree student, keen on Cyber and Network Security. Innovator mindset. Sportsperson, KH passionate and Fantasy book reader.

Personal Information

Berlato Stefano

Schio (VI), Italy

sb.berlatostefano@gmai.com

linkedin.com/in/berlatostefano

Education

Trento University - ongoing
Master degree in Computer Science, "ICT Innovation - Security and Privacy"

Trento University - 2014-2017
Bachelor degree in Computer Science - 110L

Work Experience

Android Reverse Engineering (Jul - Oct 2018)
2ASPIRE, Trento (Italy)
Reverse Engineering activities on Android applications. The goal is to provide a final analysis about security mechanism adopted by developers.

ICT Innovation Course: Business Idea Development "Joni" (Feb - Jun 2018)
University of Trento, Trento (Italy)
In the course context, Joni is a tool meant to help blind and visually impaired people to keep in touch with the world. My activity was the development of client-side functionalities with Python.
Project link > github.com/StefanoBerlato/Joni

University collaboration: IT assistant
University of Trento, Trento (Italy)
150 hours working contract under the "Information Systems Management" office, with the task of prepare and format excel data sheets regarding the Digital University project.

Thesis: Development of a Web-based Interface for the Orchestration of Machine Learning Components
University of Trento, Trento (Italy)
This thesis wells from the plenty of resources that the Machine Learning environment has reached nowadays, and the difficulty of integrating them

Cryptographically Enforcing of Dynamic Access Control Policies in the Cloud

Based on two papers

[1] On the Practicality of Cryptographically Enforcing Dynamic Access Control Policies in the Cloud

[2] Assisted Authoring, Analysis and Enforcement of Access Control Policies in the Cloud (SecurePG)

The Topic

On the Practicality of Cryptographically Enforcing Dynamic Access Control Policies in the Cloud

- Paper describing a scheme to cryptographically implement a RBAC0 policy.
- This can be done either with IBE/IBS or with PKI structure.
- Investigate a realistic use case (new users, permissions revocation, policy updates, ...)
- Insists on the high computational costs that this dynamic framework would require.

The Topic

Assisted Authoring, Analysis and Enforcement of Access Control Policies in the Cloud

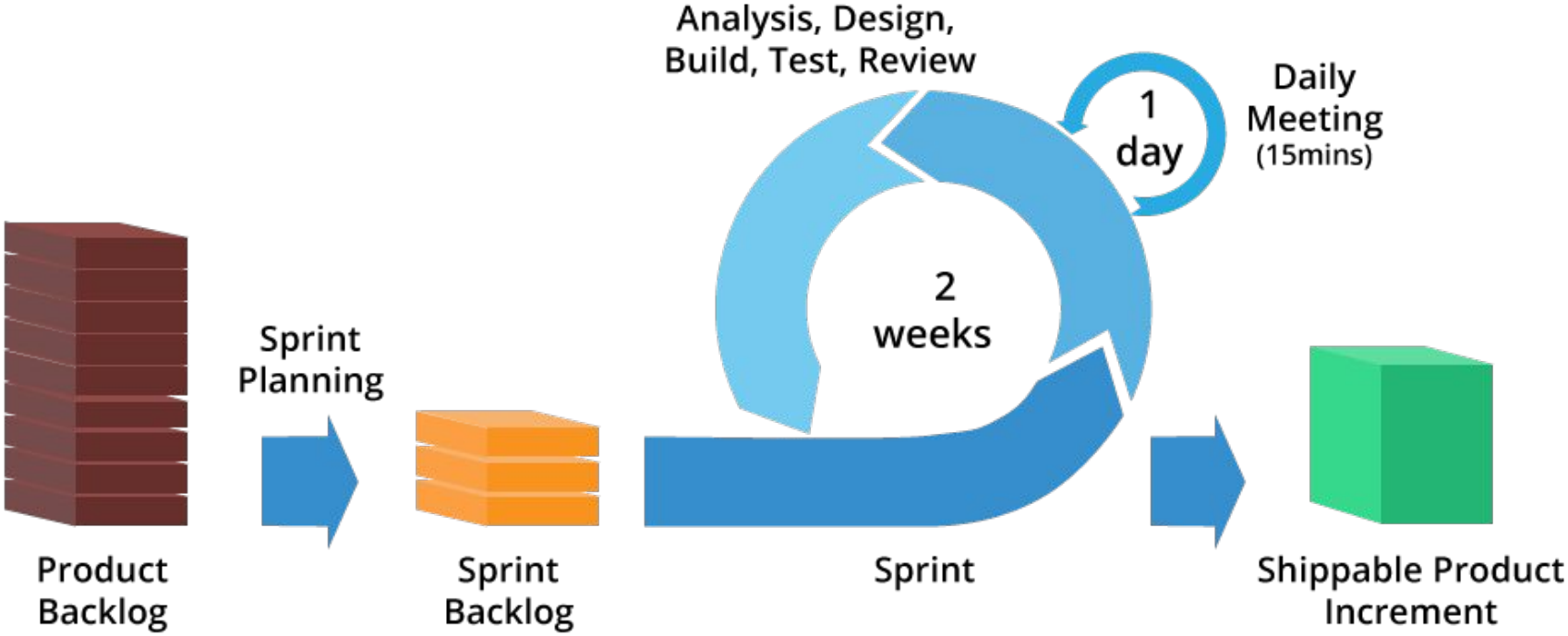
- Paper describing SecurePG, a tool developed internally by the Security&Trust unity.
- Written in Java using a MySQL database
- Allows to specify high-level language policies and then automatically translates them on different cloud platforms (AWS, Openstack).
- Note: a prototype

Integrate the cryptographic scheme described in [1] in the tools described in [2] (SecurePG).

Work divided in two main steps:

1. Implement the cryptographic AC scheme
2. Integration with SecurePG

Iterative Scrum-like process



My Contribution

First Step

$addU(u)$
- Add u to USERS
- Generate IBE private key $k_u \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}(u)$ and IBS private key $s_u \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}(u)$ for the new user u
- Give k_u and s_u to u over private and authenticated channel

$delU(u)$
- For every role r that u is a member of:
  - $revokeU(u, r)$

$addP_u(fn, f)$
- Generate symmetric key $k \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$
- Send $\langle \mathsf{F}, fn, 1, \mathbf{Enc}_u^{\mathbf{IBE}}(f), u, \mathbf{Sign}_u^{\mathbf{IBS}} \rangle$ and $\langle \mathsf{FK}, SU, \langle fn, \mathsf{RW} \rangle, 1, \mathbf{Enc}_{id}^{\mathbf{IBE}}(k), u, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
- The R.M. receives $\langle \mathsf{F}, fn, 1, c, u, sig \rangle$ and $\langle \mathsf{FK}, SU, \langle fn, \mathsf{RW} \rangle, 1, c', u, sig' \rangle$ and verifies that the tuples are well-formed and the signatures are valid, i.e., $\mathbf{Ver}_u^{\mathbf{IBS}}(\langle \mathsf{F}, fn, 1, c, u \rangle, sig) = 1$ and $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, SU, \langle fn, \mathsf{RW} \rangle, 1, c', u \rangle, sig') = 1$.
- If verification is successful, the R.M. adds $\langle fn, 1 \rangle$ to FILES and stores $\langle \mathsf{F}, fn, 1, c, u, sig \rangle$ and $\langle \mathsf{FK}, SU, \langle fn, \mathsf{RW} \rangle, 1, c', u, sig' \rangle$

$delP(fn)$
- Remove $\langle fn, v_{fn} \rangle$ from FILES
- Delete $\langle \mathsf{F}, fn, -, -, -, - \rangle$ and all $\langle \mathsf{FK}, -, \langle fn, - \rangle, -, -, -, - \rangle$

$addR(r)$
- Add $\langle r, 1 \rangle$ to ROLES
- Generate IBE private key $k_{\langle r,1 \rangle} \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}(\langle r, 1 \rangle)$ and IBS private key $s_{\langle r,1 \rangle} \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}(\langle r, 1 \rangle)$ for role $\langle r, 1 \rangle$
- Send $\langle \mathsf{RK}, SU, \langle r, 1 \rangle, \mathbf{Enc}_{SU}^{\mathbf{IBE}}(k_{\langle r,1 \rangle}), s_{\langle r,1 \rangle} \rangle, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.

$delR(r)$
- Remove $\langle r, v_r \rangle$ from ROLES
- Delete all $\langle \mathsf{RK}, -, \langle r, v_r \rangle, -, - \rangle$
- For all permissions $p = \langle fn, op \rangle$ that r has access to:
  - $revokeP(r, \langle fn, \mathsf{RW} \rangle)$

$assignU(u, r)$
- Find $\langle \mathsf{RK}, SU, \langle r, v_r \rangle, c, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{RK}, SU, \langle r, v_r \rangle, c \rangle, sig) = 1$
- Decrypt keys $(k_{\langle r,v_r \rangle}, s_{\langle r,v_r \rangle}) = \mathbf{Dec}_{kSU}^{\mathbf{IBE}}(c)$
- Send $\langle \mathsf{RK}, u, \langle r, v_r \rangle, \mathbf{Enc}_u^{\mathbf{IBE}}(k_{\langle r,v_r \rangle}, s_{\langle r,v_r \rangle}), \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.

$revokeU(u, r)$
- Generate new role keys $k_{\langle r,v_r+1 \rangle} \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}(\langle r, v_r + 1 \rangle)$, $s_{\langle r,v_r+1 \rangle} \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}(\langle r, v_r + 1 \rangle)$
- For all $\langle \mathsf{RK}, u', \langle r, v_r \rangle, c, sig \rangle$ with $u' \neq u$ and $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{RK}, u', \langle r, v_r \rangle, c \rangle, sig) = 1$:
  - Send $\langle \mathsf{RK}, u', \langle r, v_r + 1 \rangle, \mathbf{Enc}_{u'}^{\mathbf{IBE}}(k_{\langle r,v_r+1 \rangle}, s_{\langle r,v_r+1 \rangle}), \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
- For every fn such that there exists $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, op \rangle, v_{fn}, c, SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, \langle r, v_r \rangle, p, v_{fn}, c, SU \rangle, sig) = 1$:
  - For every $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, op \rangle, v, c', SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, op' \rangle, v, c', SU \rangle, sig) = 1$:
    - Decrypt key $k = \mathbf{Dec}_{k_{\langle r,v_r \rangle}}^{\mathbf{IBE}}(c')$
    - Send $\langle \mathsf{FK}, \langle r, v_r + 1 \rangle, \langle fn, op' \rangle, v, \mathbf{Enc}_{\langle r,v_r+1 \rangle}^{\mathbf{IBE}}(k), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
  - Generate new symmetric key $k' \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$ for p
  - For all $\langle \mathsf{FK}, id, \langle fn, op' \rangle, v_{fn}, c'', SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, id, \langle fn, op' \rangle, v_{fn}, c'', SU \rangle, sig) = 1$:
    - Send $\langle \mathsf{FK}, id, \langle fn, op' \rangle, v_{fn} + 1, \mathbf{Enc}_{id}^{\mathbf{IBE}}(k'), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
  - Increment v_{fn} in FILES, i.e., set $v_{fn} := v_{fn} + 1$
- Increment v_r in ROLES, i.e., set $v_r := v_r + 1$
- Delete all $\langle \mathsf{RK}, -, \langle r, v_r \rangle, -, - \rangle$
- Delete all $\langle \mathsf{FK}, \langle r, v_r \rangle, -, -, -, -, - \rangle$

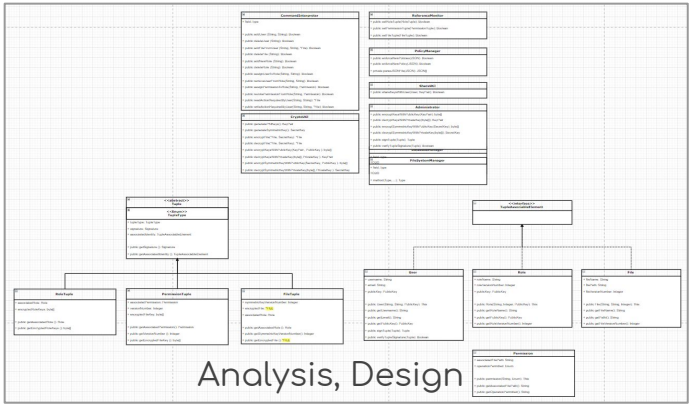
$assignP(r, \langle fn, op \rangle)$
- For all $\langle \mathsf{FK}, SU, \langle fn, \mathsf{RW} \rangle, v, c, id, sig \rangle$ with $\mathbf{Ver}_{id}^{\mathbf{IBS}}(\langle \mathsf{FK}, SU, \langle fn, \mathsf{RW} \rangle, v, c, id \rangle, sig) = 1$:
  - If this adds Write permission to existing Read permission, i.e., $op = \mathsf{RW}$ and there exists $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{Read} \rangle, v, c', SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, op' \rangle, v, c', SU \rangle, sig) = 1$:
    - Send $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{RW} \rangle, v, c', SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
    - Delete $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{Read} \rangle, v, c', SU, sig \rangle$
  - If the role has no existing permission for the file, i.e., there does not exist $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, op' \rangle, v, c', SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, op' \rangle, v, c, SU \rangle, sig) = 1$:
    - Decrypt key $k = \mathbf{Dec}_{kSU}^{\mathbf{IBE}}(c)$
    - Send $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, op \rangle, v, \mathbf{Enc}_{\langle r,v_r \rangle}^{\mathbf{IBE}}(k), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.

$revokeP(r, \langle fn, op \rangle)$
- If $op = \mathsf{Write}$:
  - For all $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{RW} \rangle, v, c, SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{RW} \rangle, v, c, SU \rangle, sig) = 1$:
    - Send $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{Read} \rangle, v, c, SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
    - Delete $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{RW} \rangle, v, c, SU, sig \rangle$
- If $op = \mathsf{RW}$:
  - Delete all $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, - \rangle, -, -, - \rangle$
  - Generate new symmetric key $k' \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$
  - For all $\langle \mathsf{FK}, r', \langle fn, op' \rangle, v_{fn}, c, SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, r', \langle fn, op' \rangle, v, c, SU \rangle, sig) = 1$:
    - Send $\langle \mathsf{FK}, r', \langle fn, op' \rangle, v_{fn} + 1, \mathbf{Enc}_{r'}^{\mathbf{IBE}}(k'), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
  - Increment v_{fn} in FILES, i.e., set $v_{fn} := v_{fn} + 1$

$readU_u(fn)$
- Find $\langle \mathsf{F}, fn, v, c, id, sig \rangle$ with valid ciphertext c and valid signature sig, i.e., $\mathbf{Ver}_{id}^{\mathbf{IBS}}(\langle \mathsf{F}, fn, 1, c, id \rangle, sig) = 1$
- Find a role r such that the following hold:
  - u is in role r, i.e., there exists $\langle \mathsf{RK}, u, \langle r, v_r \rangle, c', sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{RK}, u, \langle r, v_r \rangle, c' \rangle, sig) = 1$
  - r has read access to version v of fn, i.e., there exists $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, op \rangle, v, c'', SU, sig' \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, op \rangle, v, c'', SU \rangle, sig') = 1$
- Decrypt role key $k_{\langle r,v_r \rangle} = \mathbf{Dec}_{k_u}^{\mathbf{IBE}}(c')$
- Decrypt file key $k = \mathbf{Dec}_{\langle r,v_r \rangle}^{\mathbf{IBE}}(c'')$
- Decrypt file $f = \mathbf{Dec}_k^{\mathbf{Sym}}(c)$

$writeU_u(fn, f)$
- Find a role r such that the following hold:
  - u is in role r, i.e., there exists $\langle \mathsf{RK}, u, \langle r, v_r \rangle, c, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{RK}, u, \langle r, v_r \rangle, c \rangle, sig) = 1$
  - r has write access to the newest version of fn, i.e., there exists $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{RW} \rangle, v_{fn}, c', SU, sig' \rangle$ and $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{RW} \rangle, v, c', SU \rangle, sig') = 1$
- Decrypt role key $k_{\langle r,v_r \rangle} = \mathbf{Dec}_{k_u}^{\mathbf{IBE}}(c)$
- Decrypt file key $k = \mathbf{Dec}_{\langle r,v_r \rangle}^{\mathbf{IBE}}(c')$
- Send $\langle \mathsf{F}, fn, v_{fn}, \mathbf{Enc}_k^{\mathbf{Sym}}(f), \langle r, v_r \rangle, \mathbf{Sign}_{\langle r,v_r \rangle}^{\mathbf{IBS}} \rangle$ to R.M.
- The R.M. receives r and $\langle \mathsf{F}, fn, v, c'', \langle r, v_r \rangle, sig'' \rangle$ and verifies the following:
  - The tuple is well-formed with $v = v_{fn}$
  - The signature is valid, i.e., $\mathbf{Ver}_{\langle r,v_r \rangle}^{\mathbf{IBS}}(\langle \mathsf{F}, fn, v, c'', \langle r, v_r \rangle \rangle, sig'') = 1$
  - r has write access to the newest version of fn, i.e., there exists $\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{RW} \rangle, v_{fn}, c', SU, sig' \rangle$ and $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle \mathsf{FK}, \langle r, v_r \rangle, \langle fn, \mathsf{RW} \rangle, v_{fn}, c', SU \rangle, sig') = 1$
- If verification is successful, the R.M. replaces $\langle \mathsf{F}, fn, -, -, -, - \rangle$ with $\langle \mathsf{F}, fn, v_{fn}, c'', \langle r, v_r \rangle, sig'' \rangle$

Implementation of RBAC0 using IBE and IBS given in [1]



Analysis, Design

Testing, coding

Integration with SecurePG

Questions

Thanks

Layout of the slides by Coman Catalin Andrei