

Soutenance projet annuel

Audit des implantations SSL/TLS

Claire Smets – William Boisseleau – Pascal Edouard –
Mathieu Latimier – Julien Legras

Master 2 Sécurité des Systèmes Informatiques

28/02/2014



Sujet et problématique I

Motivations

- incertitudes cryptographiques liées aux récents scandales ;
- 2012 – étude de l'Université du Michigan sur la sécurité d'internet : *Mining your Ps and Qs : Widespread Weak Keys in Network Devices*

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
              // guaranteed to be random.  
}
```

Sujet et problématique II

Projet

- mesure de l'évolution par rapport à cette étude ;
- identification des problèmes avérés ;
- audit de la principale bibliothèque : OpenSSL.

Exigences du client

- audit des certificats RSA ;
- quantité importante de données pour des résultats représentatifs (500 000 certificats).

Sommaire

- 1 Audit des clefs RSA des certificats
- 2 Audit d'OpenSSL
- 3 Analyse dynamique du navigateur client
- 4 Conclusion

Récupération des adresses



ZMAP

- open source développé en 2013 par l'équipe de l'Université du Michigan ;
- scanneur de ports : jusqu'à 1,4 millions de paquets SYN par seconde.

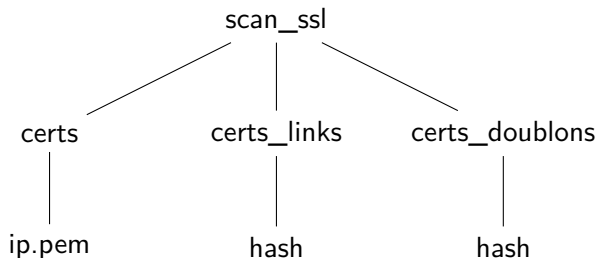
Récupération des certificats I



Récupération des certificats

- commandes openssl pour établir la connexion SSL ;
- enregistrement de la session ;
- extraction des certificats et des clefs RSA de la session SSL ;
- élimination des doublons.

Gestion des doublons



Certificats récupérés



Certificats malformés

- *Common Name* vide : validation impossible au regard du nom de domaine ;
- *Serial Number* seul : absence totale d'informations sur le certificat.

Factorisation

Rappels sur la génération des clefs RSA



Algorithme de génération de clefs RSA

- 1 générer p et q premiers et différents ;
- 2 calculer $N = p \times q$ et $\phi(N) = (p - 1) \times (q - 1)$ (indicatrice d'Euler) ;
- 3 tirer aléatoirement $e \in \mathbb{Z}/\phi(N)\mathbb{Z}$;
- 4 calculer d tel que $d \times e \equiv 1 \pmod{\phi(N)}$;
- 5 *clef publique* = N, e ; *clef privée* = p, q, d

Factorisation

Factorisation des clefs RSA



Factorisation de clefs RSA

- 1 Algorithme naïf :
 - calcul de PGCD deux à deux ;
 - complexité : $O(n!)$.
- 2 Algorithme de Bernstein *How to find smooth parts of integers ?*
 - structure arborescente ;
 - complexité : $O(n \times \log(n))$.

Factorisation

PGCD deux à deux

N_1

N_2

N_3

N_4

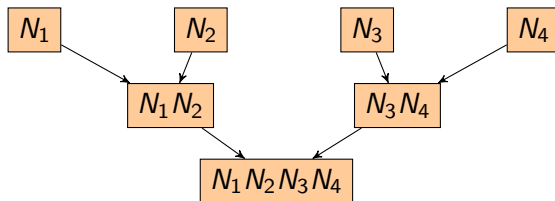
Factorisation

PGCD deux à deux



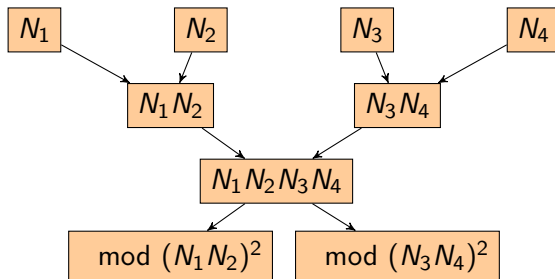
Factorisation

PGCD deux à deux



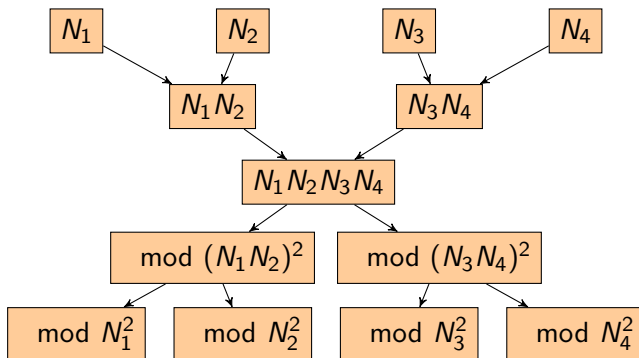
Factorisation

PGCD deux à deux



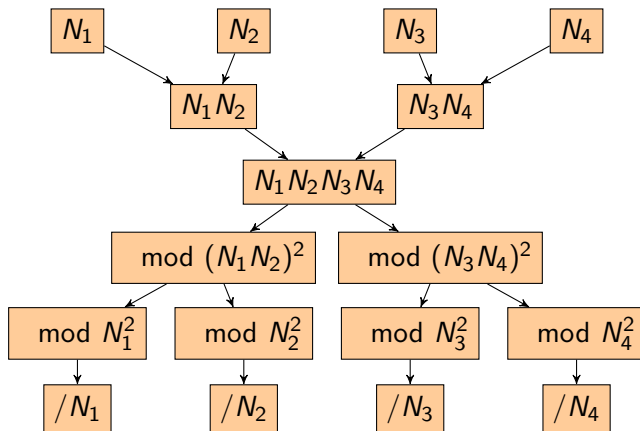
Factorisation

PGCD deux à deux



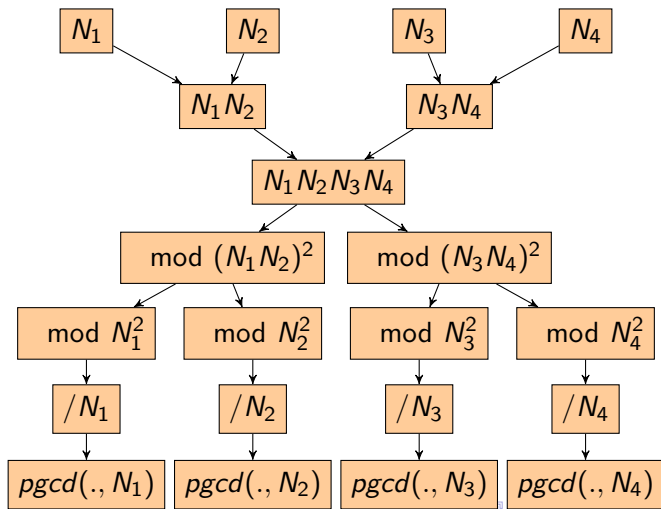
Factorisation

PGCD deux à deux



Factorisation

PGCD deux à deux



Factorisation

Exemple

6

15

77

1

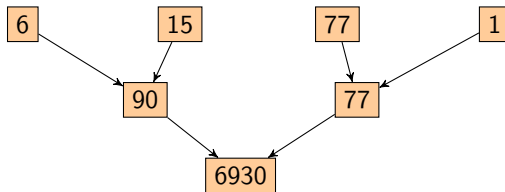
Factorisation

Exemple



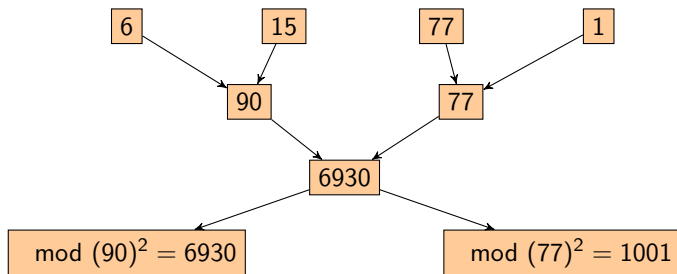
Factorisation

Exemple



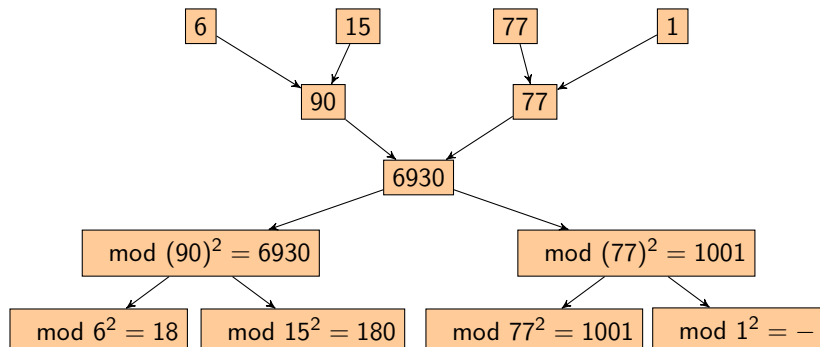
Factorisation

Exemple



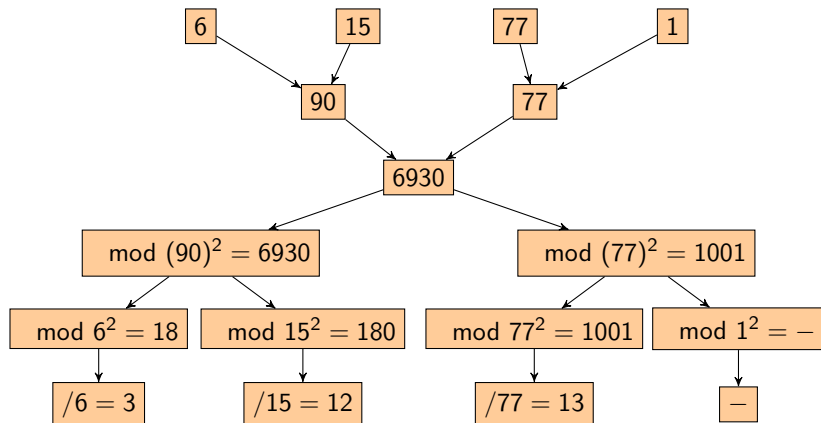
Factorisation

Exemple



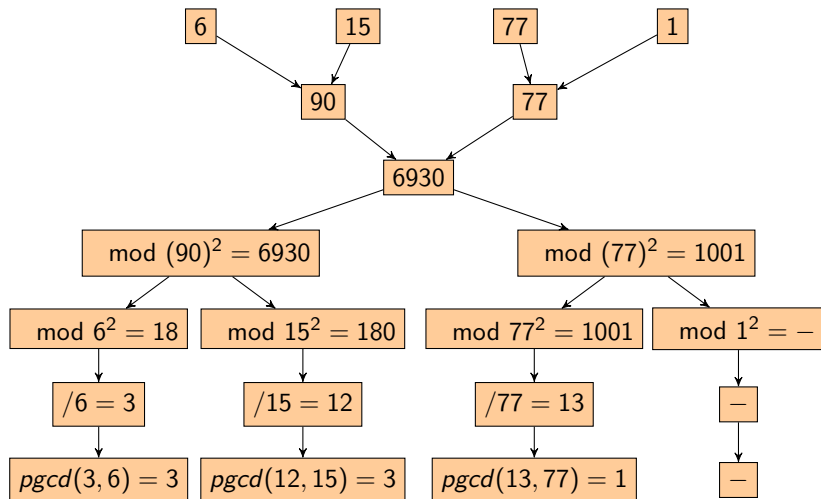
Factorisation

Exemple



Factorisation

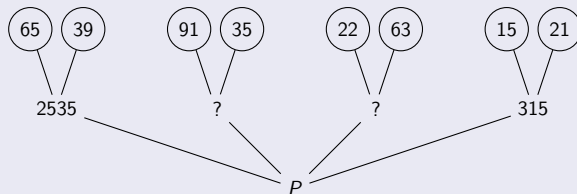
Exemple



Optimisations I

Parallélisation

En largeur avec 100 threads



Parallélisation

En hauteur : ralentissement du traitement à cause de la dépendances entre les niveaux

Optimisations II

Langage et bibliothèque

C & *libGMP*

Stockage

- programme initial : utilisation de fichiers pour stocker chaque niveau ;
- programme amélioré : stockage de l'arbre des produits en RAM (4-5 Go pour 500 000 clefs).
⇒ jusqu'à 10 fois plus rapide même avec un SSD

Calcul des carrés

Préférer la fonction d'exponentiation de GMP plutôt que la multiplication ⇒ jusqu'à 5 fois plus rapide

Factorisation

Démonstration

Jeu de données : 15, 22, 437, 1189, 527, 21, 3233, 3827

Bilan

Deux jeux de données

- 1 clefs récupérées par notre script : 85 000 ;
- 2 clefs du projet Sonar : 527 000 fin janvier
<https://scans.io/stdudy/sonar.rdns/>.

Traitement

- factorisation ;
- insertion dans une base de données ;
- développement d'une interface web avec graphiques :
 - 1 125 vulnérables (0,12%) ;
 - 2 2 382 vulnérables (0,46%) – 40 de Cisco.

Analyse d'un certificat vulnérable

SSL Report: s[REDACTED].net (8[REDACTED]3)

Assessed on: Thu Feb 27 20:19:19 UTC 2014 | [Clear cache](#)

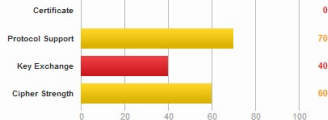
[Scan Another »](#)

Summary

Overall Rating



If trust issues are ignored: C



Documentation: [SSL/TLS Deployment Best Practices](#), [SSL Server Rating Guide](#), and [OpenSSL Cookbook](#).

This server's certificate is not trusted. Grade set to F.

This server does not mitigate the [CRIME attack](#). Grade capped to B.

The server supports only older protocols, but not the current best TLS 1.2. Grade capped to B.

The server private key is not strong enough. Grade capped to B.

There is no support for secure renegotiation. [MORE INFO »](#)

The server does not support Forward Secrecy with the reference browsers. [MORE INFO »](#)

Introduction I

OpenSSL

API libre implémentant des primitives cryptographiques et le protocole SSL/TLS.

450 000 lignes de code.

Historique d'OpenSSL

- SSLeay développé par Eric Young & Tim Hudson chez Cryptosoft ;
- 1998 – passation du projet à la communauté : OpenSSL ;
- 6 janvier 2014 – version 1.0.1f.

Introduction II

Contexte

- origine de vulnérabilité des certificats ;
- contexte actuel : sentiment d'incertitude ;
- beaucoup d'outils utilisés.

Audit d'OpenSSL

Trois grands axes :

- la génération de l'aléatoire ;
- la génération des clefs ;
- le protocole SSL/TLS.

Méthodologie de recherche de failles

Où ?

- site des CVE mitre ;
- site de Vigil@nce ;
- RFC et standards.

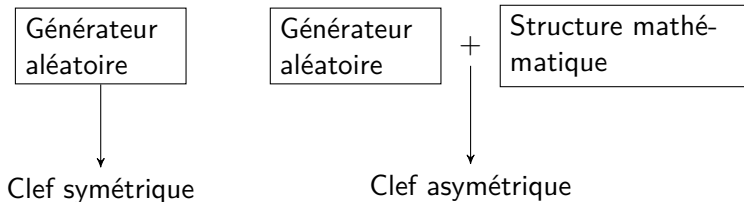
Corrections ?

Présence de corrections ? Si oui, lesquelles et par qui ?

Génération des clefs

Principes de Kerckhoffs

- le *secret* réside dans la clef ;
- les algorithmes de génération de clefs ne doivent donner aucune information sur la clef.



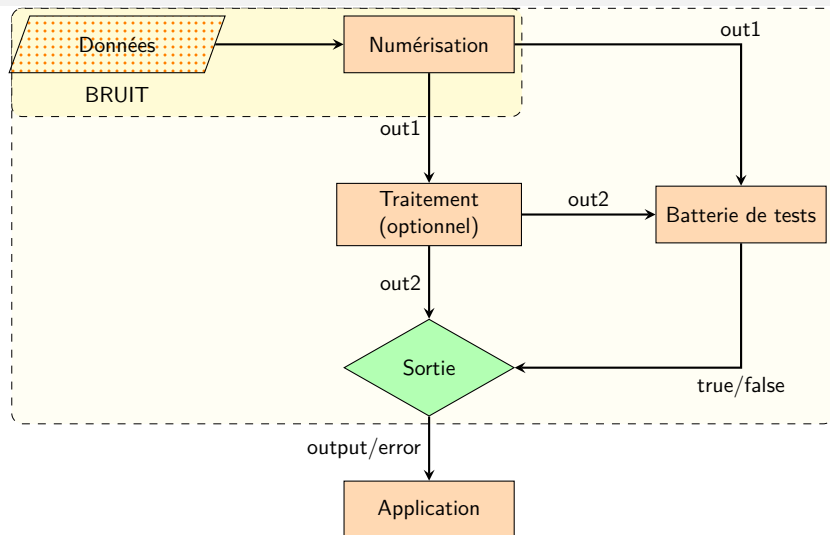
Générateur aléatoire

Définitions

Générateur

- problème aléatoire ;
- mesure de l'entropie :
$$H(X) = - \sum_x P(X = x) \times \log_2(P(X = x))$$
 (exprimée en Shannon) :
 - si la VA est équirépartie alors l'entropie est maximale ;
 - sinon inférieure à 50%.
- entropie et moduli.

Entropie



Entropie

Tests

- tests trop anciens, FIPS 140-1 (1994), non mis à jour :
 - test monobit ;
 - test poker ;
 - test runs ;
 - test long runs.

↪ adaptation des tests obsolètes (20 000 à 8 000 000 bits)
développés par l'équipe d'OpenSSL ;

- autres tests proposés dans le rapport (NIST).

Entropie – Démonstration

Faible générateur d'aléatoire sur clefs RSA

- version : OpenSSL 0.9.8 (2008) sous Debian 4.0 ;
- but : retrouver la clef privée d'un certificat ;
- moyens : bibliothèque OpenSSL, openssl-vulnkey, liste des 250 000 facteurs premiers possibles ;
- cause : suppression d'une grande source d'entropie dans crypto/rand/md_rand.c ;
- impactés : IBM & CISCO (sur le premier scan).

Faibles similaires – 2013

- Android / Linux Mint Debian Edition
- NetBSD 6.0 et OpenSSH

Génération des clefs

Attaque Man in the middle sur Diffie-Hellman

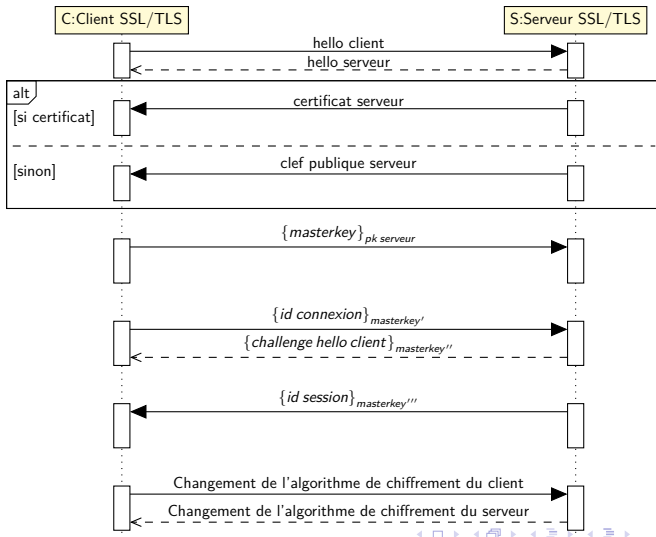
- but : déchiffrer les messages entre un client et un serveur en forçant la génération d'un secret DH ;
- moyens :
 - mode FIPS activé sous OpenSSL ;
 - écoute active sur le réseau.
- cause : génération d'un faux positif lors d'un test dans `crypto/dh/dh_key.c`

Chiffrement

Timing attack sur RSA-OAEP

- version : OpenSSL 1.0.0 (2010) ;
- but : retrouver la clef privée d'un certificat en contrôlant la taille des paramètres à hacher ;
- moyens : RSA-OAEP, variations de délais ;
- impact :
 - théorique sur les serveurs ;
 - pratique sur les systèmes embarqués.

Protocole SSL/TLS I



Protocole SSL/TLS II

Historique

- SSL 1.0 et 2.0 (1995) conçues par Netscape, 3.0 (1996) par l'IETF ;
- TLS 1.0 (1999) : légère amélioration de SSL 3, ajout d'extensions ;
- TLS 1.1 (2006) : protection contre les attaques contre CBC ;
- TLS 1.2 (2008) : remplacement MD5/SHA1 par SHA256, ajout des modes GCM et CCM.

Failles notables d'OpenSSL

SSL 3 et TLS 1.0 :

- 2002 - attaque sur le padding du mode CBC par Serge Vaudenay (reprise par Lucky Thirteen) : *timing attack* ;

TLS 1.1 :

- **2011** (CVE-2011-4576) - récupération d'informations sur un échange précédent : mémoire non réinitialisée ;
- **2013** (CVE-2013-0169)- Lucky Thirteen : *timing attack*, 2^{23} sessions TLS pour retrouver un bloc en clair ;

TLS 1.2 :

- **2013** (CVE-2013-6449) - déni de service en envoyant une structure mal formée causant un *crash* dans la fonction `ssl_get_algorithm2`

Attaque à clair choisi en man in the middle

Attaque à clair choisi en man in the middle

- versions : toutes ;
- but : déchiffrement du cookie de session ;
- moyens : *cipersuite* en TLS 1.0 ou SSL 3.0 en mode de chiffrement CBC ;
- logiciel : BEAST.

Remarques

Points négatifs

- documentation peu fournie ;
- développement en mode réactif ;
- illisibilité du code :

```
n=((p[0]&0x7f)<<8)|p[1];
```

Points positifs

- nom des fichiers standardisé : s3_ ;
- commentaires des commits.

Ouverture

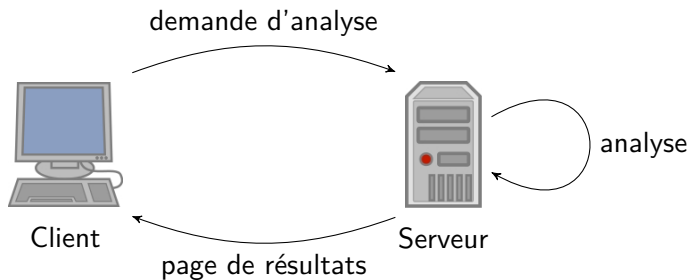
Alternatives

- CyaSSL ;
- GnuTLS ;
- MatrixSSL ;
- Network Security Services (Firefox) ;
- PolarSSL (pas de support du DTLS) ;
- Java Secure Socket Extension (pas de support du DTLS).

Attention – 25 février 2014

CVE-2014-1959 dans GnuTLS : validation de certificat (X509 version 1) intermédiaire comme un certificat d'AC par défaut.

Faiblesses identifiées I



Faiblesses identifiées II

Critères

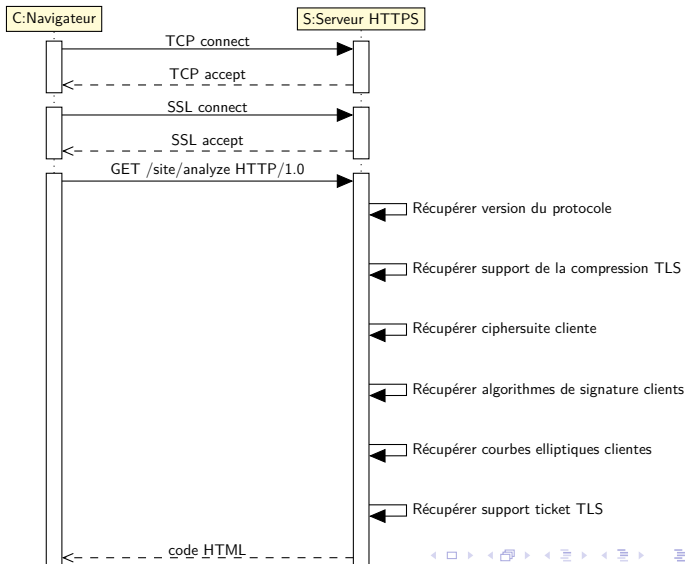
- version du protocole ;
- *ciphersuites* proposées par le client ;
- courbes elliptiques supportées ;
- algorithmes de signature ;
- compression TLS ;
- activation ou non du ticket de session.

Implémentation I

Architectures possibles

- 1 journalisation de la connexion HTTPS avec une sonde IDS puis récupération des informations avec un langage tel que PHP → couche réseau ;
- 2 serveur HTTPS avec *libssl* puis analyse de la structure de la session → couche applicative.

Implémentation II



Démonstration

Analyse de la sécurité des navigateurs clients

- Sous le navigateur graphique Chrome : le plus utilisé dans le monde.
- Sous le navigateur console lynx : apprécié chez les développeurs.

Modification manuelle

Nous pouvons modifier la *ciphersuite* du client avec la commande `s_client`.

Conclusion

Bilan du projet – <https://github.com/RandomGuys/>

- Appli RC : récupération de certificats RSA ;
- Appli F : factorisation de clefs RSA ;
- site web :
 - statistiques ;
 - analyse dynamique navigateur.
- rapport d'audit.

Apports du projet

- recherche approfondie de normes ;
- riche en développement et en recherche.

Questions ?