Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PREMIER MINISTRE

Secrétariat général de la défense nationale

Agence nationale de la sécurité des systèmes d'information

Rapport de certification ANSSI-CSPN-2009/06

Paris, le 21 Décembre 2009

Le directeur général de l'agence nationale de la sécurité des systèmes d'information

Patrick Pailloux
[ORIGINAL SIGNE]

SÉCURITÉ CERTIFICATION

Avertissement

Ce rapport est destiné à fournir aux commanditaires un document leur permettant d'attester du niveau de sécurité offert par le produit dans les conditions d'utilisation ou d'exploitation définies dans ce rapport pour la version qui a été évaluée. Il est destiné également à fournir à l'acquéreur potentiel du produit les conditions dans lesquelles il pourra exploiter ou utiliser le produit de manière à se trouver dans les conditions d'utilisation pour lesquelles le produit a été évalué et certifié. C'est pourquoi ce rapport de certification doit être lu conjointement aux guides d'utilisation et d'administration évalués ainsi qu'à la cible de sécurité du produit qui décrit les menaces, les hypothèses sur l'environnement et les conditions d'emploi présupposées afin que l'utilisateur puisse juger de l'adéquation du produit à son besoin en termes d'objectifs de sécurité.

La certification ne constitue pas en soi une recommandation du produit par l'Agence nationale de la sécurité des systèmes d'information (ANSSI), et ne garantit pas que le produit certifié soit totalement exempt de vulnérabilités exploitables.

Toute correspondance relative à ce rapport doit être adressée au :

Secrétariat général de la défense nationale
Agence nationale de la sécurité des systèmes d'information
Centre de certification
51, boulevard de la Tour Maubourg
75700 Paris cedex 07 SP

certification.anssi@ssi.gouv.fr

Référence du rapport de certification

ANSSI-CSPN-2009/06

Nom du produit

Bro v1.4

Référence/version du produit

Version 1.4

Critères d'évaluation et version

CERTIFICATION DE SECURITE DE PREMIER NIVEAU (CSPN, Phase expérimentale)

Développeur(s)

Lawrence Berkeley National Laboratory University of California, Berkeley USA

http://bro-ids.org

Commanditaire

Agence nationale de la sécurité des systèmes d'information

Secrétariat Général de la Défense Nationale
51, boulevard de la Tour Maubourg
75700 – Paris – 07 SP
France

Centre d'évaluation

Amossys

4 bis, allée du Bâtiment 35000 Rennes, France

Tél : +33 (0)2 99 23 15 79, mél : frederic.remi@amossys.fr

Préface

La certification

La certification de la sécurité offerte par les produits et les systèmes des technologies de l'information est régie par le décret 2002-535 du 18 avril 2002, publié au Journal officiel de la République française. Ce décret indique que :

- L'agence nationale de la sécurité des systèmes d'information élabore les **rapports de certification**. Ces rapports précisent les caractéristiques des objectifs de sécurité proposés. Ils peuvent comporter tout avertissement que ses rédacteurs estiment utile de mentionner pour des raisons de sécurité. Ils sont, au choix des commanditaires, communiqués ou non à des tiers ou rendus publics (article 7).
- Les **certificats** délivrés par le Premier ministre attestent que l'exemplaire des produits ou systèmes soumis à évaluation répond aux caractéristiques de sécurité spécifiées. Ils attestent également que les évaluations ont été conduites conformément aux règles et normes en vigueur, avec la compétence et l'impartialité requises (article 8).

Les procédures de certification sont disponibles sur le site Internet www.ssi.gouv.fr.

Table des matières

1 Le produit

1.1 Présentation du produit

Le produit évalué est Bro v1.4 **,** développé par le Lawrence Berkeley National Laboratory University of California, Berkeley USA .

Bro est un système de détection d'intrusion réseau (« Network Intrusion Detection System ») *open source*, disponible pour les systèmes d'exploitation de type Unix (dont Linux, FreeBSD et OpenBSD), qui analyse le trafic réseau à la recherche de toute activité suspecte (caractéristique d'une attaque ou d'une violation de la politique de sécurité en vigueur sur le réseau surveillé). L'analyse se fait de manière passive : dans sa configuration par défaut, Bro n'altère pas les paquets réseaux qu'il traite[1].

Bro détecte les intrusions en trois étapes :

- la première consiste à capter le trafic réseau et à décoder les différentes couches protocolaires (de manière à en extraire la sémantique applicative). Cette étape fournit des événements de « haut niveau » qui pourront par la suite être analysés ;
- la seconde (réalisée au cours du déroulement de la première étape) consiste à vérifier la présence de motifs, qui constituent des signatures d'attaques, dans la charge des paquets IP (ou du flux TCP si le ré-assemblement de flux TCP est activé) ou de certains champs des protocoles applicatifs (par exemple, HTTP dans la version évaluée du produit). Des événements sont générés en cas de concordance ;
- la troisième étape consiste à analyser les événements générés lors des deux étapes précédentes par des scripts d'analyse. Cette analyse permet à la fois la détection d'attaques connues au préalable (qui sont décrites en termes de signatures ou d'événements) et d'anomalies (par exemple, la présence de connexions de certains utilisateurs vers certains services ou l'occurrence de tentatives de connexions infructueuses).

Le produit comprend une base de signatures minimaliste. Cette base de signatures, qui n'est plus maintenue et date d'octobre 2003, provient de l'adaptation de la conversion d'une base de signatures fournie avec le produit Snort[2]. Cependant cette base n'est proposée qu'à titre d'exemple. La rédaction des règles est donc laissée à la charge de l'administrateur de Bro.

[1] Il est toutefois possible d'utiliser des scripts externes à Bro pour bloquer des actions suspectes. Ce n'est cependant pas le mode de fonctionnement standard de Bro.

[2] Voir le fichier « `ex.web-rules.sig` », placé dans le répertoire « `policy/sigs/` » de la distribution de Bro, qui indique que la conversion est issue du fichier « `snortrules-current.tar.gz` » de Snort à la date du 9 octobre 2003.

1.2 Description du produit évalué

La cible de sécurité [ST] définit le produit évalué, ses fonctionnalités de sécurité évaluées et son environnement d'exploitation.

1.2.1 Catégorie du produit

1 - détection d'intrusion
2 - anti-virus, protection contre les codes malicieux
3 – pare-feu
4 - effacement de données
5 - administration et supervision de la sécurité
6 - identification, authentification et contrôle d'accès
7 - communication sécurisée
8 - messagerie sécurisée
9 - stockage sécurisé
10 - matériel et logiciel embarqué
99-Autres

1.2.2 Identification du produit

Remarque :
Le numéro de version du produit est disponible à plusieurs emplacements :

- dans le nom de l'archive de Bro « *bro*-**1.4**-*release.tar.gz* » ;
- au sein de la distribution de Bro, dans le fichier `VERSION` ;
- en utilisant Bro avec l'option « `--version` » (voir la figure 1).

```
bro@cspn-bro:~$ /usr/local/bro/bin/bro --version
/usr/local/bro/bin/bro version 1.4
bro@cspn-bro:~$
```

Figure 1 – Obtenir la version de Bro

1.2.3 Services de sécurité

La principale fonction du produit Bro est la **détection d'intrusion réseau**. Il s'agit essentiellement d'une sonde de détection d'intrusion comprenant :

- un **capteur** chargé de capturer les paquets réseau et de décoder les protocoles ;
- d'un **analyseur** reposant sur :
  - un mécanisme de **reconnaissance de motifs**, utilisant des **signatures** afin de détecter des attaques dans le trafic réseau ;
  - un mécanisme de **gestion des événements** (produits par le capteur ou le mécanisme de reconnaissance de signatures) à l'aide de **scripts** écrits dans le « langage Bro ».

Le produit fournit des exemples de scripts et de signatures. L'efficacité du service rendu repose donc sur la spécification explicite de la part de l'utilisateur (ou d'un tiers de confiance) de signatures et de scripts adéquats.

1.2.4 Configuration évaluée

La configuration évaluée ne couvre pas la fonction de gestion des alarmes.

2 L'évaluation

2.1 Référentiels d'évaluation

L'évaluation a été menée conformément au référentiel « Certification de Sécurité de Premier Niveau en phase expérimentale ». Les références des documents se trouvent en annexe 2.

2.2 Charge de travail prévue et durée de l'évaluation

La charge de travail prévue lors de la demande de certification était conforme à la charge de travail préconisée dans [CSPN] pour un produit ne comportant pas de mécanismes cryptographiques, soit 20 hommes x jours. L'évaluation s'est déroulée au cours du mois de septembre 2009.

2.3 Travaux d'évaluation

Ce paragraphe apporte des compléments sur la cible de sécurité [ST] fournie en entrée de l'évaluation. Ces précisions sont issues du [RTE] élaboré par l'évaluateur suite à ses travaux.

2.3.1 Fonctionnalités, environnement d'utilisation et de sécurité

2.3.1.1 Spécification de besoin du produit

Conforme à [ST].

2.3.1.2 Biens sensibles manipulés par le produit

Conforme à [ST].

2.3.1.3 Description des menaces contre lesquelles le produit apporte une protection

Conforme à [ST]. La menace M6 est couverte par l'environnement technique et organisationnel mis en place pour évaluer le logiciel.

2.3.1.4 Fonctions de sécurité

Conforme à [ST].

2.3.1.5 Utilisateurs typiques

Conforme à [ST].

2.3.2 Installation du produit

2.3.2.1 Plate-forme de test

Bro est installé au sein d'une plate-forme de test sur laquelle il est possible de simuler du trafic réseau (de fond, d'attaque et de stress). La figure 2 schématise cette plate-forme.

Cette plate-forme comprend trois systèmes physiques différents :

- La machine IDS comprend un système OpenBSD sur lequel le logiciel Bro est installé. Il s'agit d'une *machine dédiée* dont les spécifications[1] permettent, d'après la documentation de Bro[2], de traiter des données provenant d'un réseau Gigabit (nombre de paquets par seconde inférieur à 50 000).
- La machine serveur est un système simulant une DMZ (zone démilitarisée) dans laquelle sont installés les différents services (serveur Web, serveur de courriels et serveurs de noms de domaines). Les différents serveurs sont placés dans des machines virtuelles VirtualBox différentes.
- La machine de génération de trafic permet de générer le trafic (légitime, de stress et d'attaque) observé par l'IDS et à destination des serveurs de la DMZ.

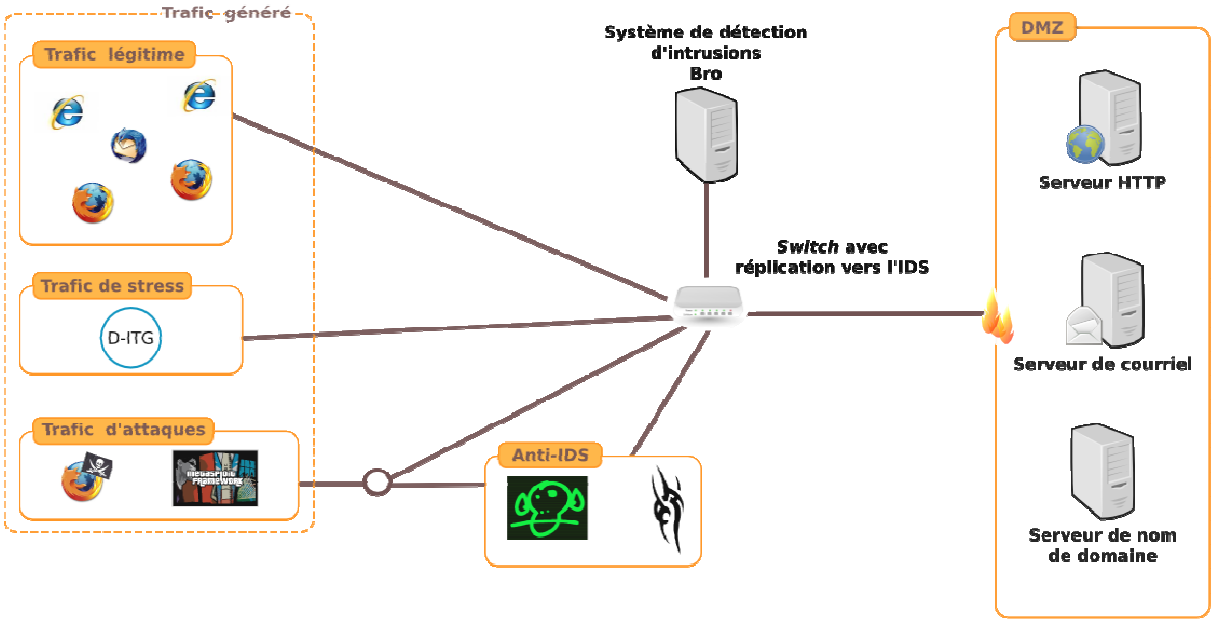

Figure 2 – Plate-forme de test au sein de laquelle Bro a été évalué.

2.3.2.2 Particularités de paramétrage de l'environnement

Bro est installé sur le système d'exploitation OpenBSD dans sa version 4.5. Le système est installé à partir du jeu de CD-ROM officiel et ne présente pas de particularité de configuration. Aucun patch ou correctif n'a été appliqué sur l'OS. Certains logiciels et bibliothèques, nécessaires au bon fonctionnement de Bro, ont été installés à partir des paquets binaires[3] officiels d'OpenBSD.

[1] Il s'agit d'une machine disposant d'un processeur « Intel Core2 Duo » (2.6 GHz) et ainsi que de 6 Go de mémoire vive. Elle comprend une carte réseau NETGEAR Gbit dédiée à la capture de trafic.
[2] Voir la section « *Requirements* » de la documentation utilisateur présente sur le Wiki.
[3] Il s'agit d'un moyen permettant de récupérer et d'installer de manière simple les logiciels sur le système d'exploitation OpenBSD.

Les dépendances obligatoires et certaines dépendances facultatives de Bro[1] ont été installées explicitement sur le système d'exploitation :

- GeoIP-1.4.5p0 ;
- bison-2.3 ;
- libmagic-4.26.

Certaines dépendances obligatoires n'ont pas nécessité d'installation explicite car elles sont fournies par défaut par le système OpenBSD (notamment la bibliothèque `libpcap`).

2.3.2.3 Options d'installation retenues pour le produit

Aucune option particulière n'a été spécifiée durant la phase d'installation (compilation et déploiement).

La configuration de base de Bro a été réalisée grâce au script *BroLite*, dont les lacunes ont été corrigées selon les informations fournies dans la documentation utilisateurs ([Guide]).

Le fichier de politique de départ utilisé (définie par la variable `BRO_START_POLICY` dans le fichier de configuration `bro.rc`) est le suivant :

```
# Chargement de la configuration standard.
@load brolite

# Activation du fichier d'alertes (fichier alarm.log).
@load alarm

# Émission d'une alerte lors de la vérification d'une signature.
@load signatures

# Vérification complète des paquets.
redef dpd_match_only_beginning = F;

# Définition du réseau local.
redef local_nets: set[subnet] = {
   192.168.0.0/16,
};
```

2.3.2.4 Description de l'installation et des non-conformités éventuelles

Le produit a été installé à partir des sources contenues dans une archive. Cette archive a été téléchargée à partir du site officiel du projet[2]. Les différentes étapes de l'installation (compilation, déploiement, configuration) ont été réalisées en suivant les instructions fournies par la documentation ([Guide]). La figure 3 illustre le résultat du script de configuration par défaut utilisé.

[1] Voir le document [Livrable2] section 5 (« Pré-requis »).
[2] Depuis la page http://bro-ids.org/download.html ou directement à l'adresse ftp://bro-ids.org/bro-1.4-release.tar.gz.

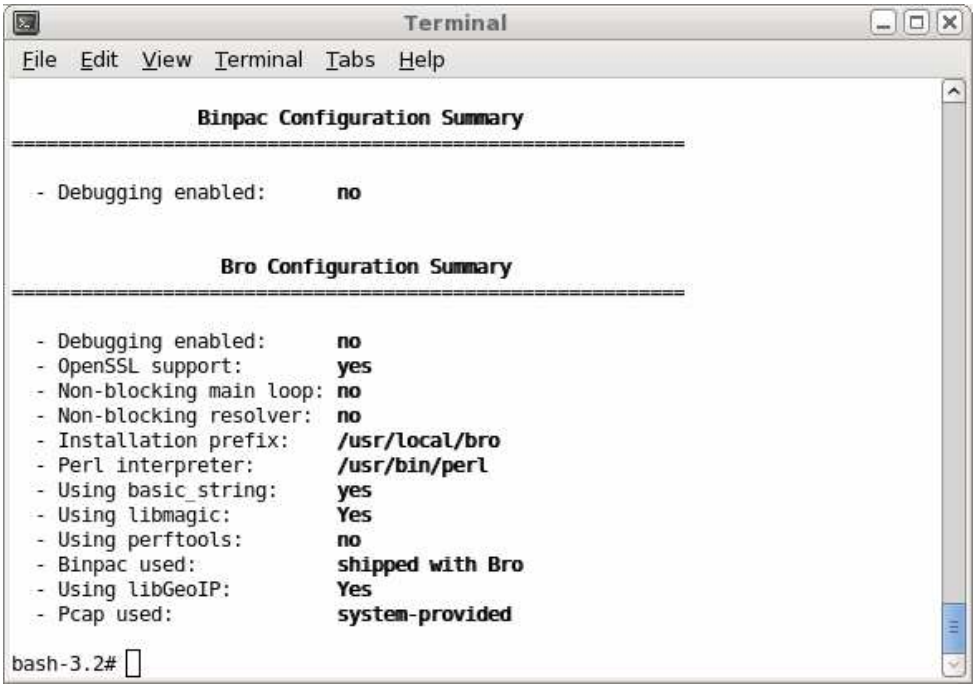

Figure 3 – Résultat des options de compilation (« `./configure` »)

La phase de configuration initiale a nécessité une procédure particulière non documentée officiellement. En effet, la version 1.4 de Bro, utilisée dans le cadre de cette évaluation, présente certaines régressions par rapport à la version précédente. Le script de configuration « *BroLite* » ne fonctionne pas avec la version 1.4 de Bro et pour le système OpenBSD.

Il est préférable d'appliquer un correctif non officiel disponible sur le système de suivi de bogues pour faire fonctionner *BroLite*.

Certains tests ont été menés sur une version recompilée de Bro avec les options d'aide au déboguage (« `--enable-debug` »).

2.3.2.5 Durée de l'installation

L'installation de base du logiciel Bro, sur un système OpenBSD préalablement installé et configuré, dure environ 1h.

2.3.2.6 Notes et remarques diverses

La configuration du produit Bro dans l'environnement spécifié par la cible de sécurité (système OpenBSD) nécessite de mettre en œuvre des procédures particulières, non documentées officiellement. Les différentes étapes nécessaires pour finaliser l'installation sont mentionnées dans le document [Guide].

2.3.3 Analyse de la conformité

2.3.3.1 Analyse de la documentation

La distribution de Bro 1.4 officielle intègre différents documents. Ceux-ci sont présents dans le répertoire « `doc/` » une fois l'archive décompressée. Il s'agit cependant de documents qui sont pour la plupart obsolètes[1]. La présence de ces fichiers introduit un risque de confusion pour l'utilisateur final de Bro.

Les versions actualisées sont disponibles sur le site de Bro, dans un espace de type Wiki[2]. L'évaluation s'est appuyée sur les versions des documents fournies sur le Wiki. Ces documents sont rédigés en anglais.

Certaines fonctionnalités ne sont pas décrites dans la documentation mais font l'objet d'une page dédiée du Wiki (par exemple, la documentation du mécanisme de détection dynamique de protocole). Les diapositives des présentations faites lors de travaux dirigés (« *Workshop* ») sont accessibles http://www.bro-ids.org/bro-workshop-2009-2/

La documentation utilisateur officielle de Bro, [DOC1], s'avère être largement incomplète et parfois obsolète car n'intégrant pas les dernières modifications apportées à Bro, en particulier en ce qui concerne l'installation. En effet, le système d'installation et de configuration initial de Bro est en phase transitoire, passant de « *BroLite* » à « *BroControl* »[3]. Il n'est pas possible d'installer la version 1.4 en utilisant le système historique « *BroLite* » alors que la documentation ne fait état que de ce dernier. Il est nécessaire, soit d'installer un correctif, soit de s'appuyer sur des retours d'expériences non officiels. Par ailleurs, la documentation officielle n'explique pas comment configurer manuellement Bro. La version évaluée ne supporte pas le nouveau système de configuration (« *BroControl* »). Le document [Guide] précise la marche à suivre pour finaliser manuellement l'installation.

Le manuel de référence officiel de Bro [DOC2] présente la liste complète des fonctions et des modules disponibles dans le langage de politique Bro. Cette documentation se veut exhaustive et a pour but de documenter de manière précise le rôle de chaque élément. L'évaluation a révélé que cette documentation est obsolète sur certains points. En particulier, la liste des événements générés par les analyseurs protocolaires et la définition de la sémantique de ces événements ne reflètent pas l'état actuel de Bro. Par exemple, la documentation ne mentionne pas l'événement TCP « `connexion_EOF` », la description des événements HTTP est incomplète et incorrecte (l'événement « `http_reply` » est implémenté dans la version évaluée de Bro contrairement à ce qui est annoncé dans la documentation), etc.

Le manuel de référence fournit une description complète et à jour de la grammaire du langage de script Bro et des fonctions natives. Par contre, certaines fonctionnalités sont peu ou non documentées (par exemple, l'analyseur ICMP, le script de détection des portes dérobées, etc.). Ceci apparait dès la lecture du document qui n'est pas finalisé (chapitres vides), ce qui a été confirmé durant l'évaluation.

[1] Le répertoire « doc/ » de l'archive n'a pas été actualisé depuis plus de 2 ans.
[2] Un Wiki est un espace Web collaboratif de rédaction.
[3] Système anciennement connu sous le nom « *ClusterShell* », récemment renommé en « *BroControl* ».

Le fichier « CHANGES » [DOC3] fournit une description précise des évolutions de Bro. Ce fichier est régulièrement mis à jour.

Conclusion : la documentation officielle présente des lacunes importantes. Elle ne suffit pas à elle seule pour réaliser l'installation et la configuration de base d'un système Bro fonctionnel. Toutefois, la traduction de la documentation utilisateurs ([Guide]) contient les informations manquantes permettant d'installer Bro correctement.

2.3.3.2 Revue du code source

Globalement, le code source des différents mécanismes de Bro (capture et décodage des protocoles, gestion des signatures, gestion des scripts de politiques, etc.) respecte les bonnes pratiques de développement. La conception de certains mécanismes (notamment la gestion des signatures) traduit la prise en compte des problématiques de robustesse.

En revanche, la structuration du code source pourrait être améliorée (beaucoup de fichiers de nature différente sont situées dans le même répertoire). Elle ne facilite pas la compréhension du code. De plus, celui-ci souffre d'un manque de documentation destinée aux développeurs. Certaines parties sont peu ou pas documentées (par exemple la construction de l'arbre de signatures). Il manque un document présentant l'architecture globale de Bro.

2.3.3.3 Fonctions testées

L'analyse de la conformité a consisté à réaliser des tests fonctionnels complétés par l'analyse du code source sur les fonctions sensibles de l'IDS.

Les fonctions suivantes ont été testées :

- le module de capture et de décodage des protocoles : tests de montée en charge et tests de la conformité des protocoles réseaux standards (TCP, UDP et HTTP) ;
- le module d'analyse des événements : vérification que les événements que Bro est censé être en mesure de lever, au travers des analyseurs réseaux, le sont effectivement ;
- le mécanisme de signatures : vérification que le mécanisme de signature fonctionne tel qu'annoncé ;
- le mécanisme d'envoi des alertes au format IDMEF.

Remarque : la version évaluée de Bro ne permet pas de convertir correctement les dernières signatures issues du logiciel Snort.

2.3.3.4 Fonctions non testées

La fonction de gestion des alertes n'a pas été testée.

2.3.3.5 Synthèse des fonctions testés / non testées et des non-conformités

Fonctions testées	Résultat
Capture et décodage de protocole Analyse des événements	**Réussite**[1]
Capture et décodage de protocole (TCP)	**Réussite**
Capture et décodage de protocole (UDP)	**Réussite**
Capture et décodage de protocole (HTTP)	**Réussite**
Mécanisme de vérification de signatures	**Réussite**
Envoi des alertes au format IDMEF	**Echec**

2.3.3.6 Avis d'expert sur le produit

Les tests réalisés ont ainsi montré que Bro était en mesure de décoder les protocoles réseaux correctement pour peu que les paquets réseaux soient valides. Il présente cependant certaines faiblesses notamment dans son analyse peu rigoureuse du protocole TCP et des cas invalides en général. Il devient alors possible de tromper le moteur d'analyse du protocole TCP. Ce problème, complexe, pose la question de l'environnement dans lequel est déployée la sonde de détection. Il semble en effet indispensable de placer cette dernière derrière un pare-feu ou un élément de normalisation réseau.

Les différents tests réalisés sur le mécanisme de gestion de signature sont globalement concluants. Les non conformités identifiées ont un impact faible. Par exemple, il n'y a pas d'autres alternatives que d'utiliser des signatures simples de type *payload* pour rechercher un motif dans les en-têtes http. Cette limitation entraine la perte des avantages liés à l'utilisation d'un analyseur haut niveau, effectuant des traitements de décodage des chaines de caractères par exemple.

En ce qui concerne le respect des standards, Bro présente quelques limitations. Il n'est pas en mesure d'émettre ses alertes au format standardisé IDMEF, il faut en effet disposer d'une version modifiée de la bibliothèque `libidmef`, non distribuée avec Bro. Il semble donc difficile d'intégrer Bro dans un environnement hétérogène et de réaliser du partage d'informations avec d'autres outils du marché comme Snort ou Prelude IDS. Ceci peut pourtant être nécessaire, par exemple dans le cadre de la corrélation.

[1] Les résultats sont à pondérer car ils dépendent fortement de l'environnement et de la méthode de génération de trafic.

2.3.4 Analyse de la résistance des mécanismes et des fonctions

2.3.4.1 Liste des fonctions testées et résistance

Les mécanismes mis en jeu sont :

- la capacité de détection de Bro en présence de trafic de fond légitime ;
- la résistance de Bro aux techniques d'évasion (cette étape a également permis d'identifier des vulnérabilités de Bro, permettant à un attaquant d'échapper à la surveillance de l'IDS) ;
- l'efficacité du traitement des signatures (l'évaluation du passage à l'échelle) ;
- la gestion des défaillances.

Capacité de détection en présence de trafic de fond

Afin de tester ses capacités de détection, Bro a été testé en présence d'un trafic d'attaques, dans un environnement vulnérable.

Les attaques retenues dans le cadre de cette évaluation sont les suivantes :

- mise en œuvre d'une attaque de type déni de service sur un serveur Apache grâce à l'outil Slowloris[1] ;
- mise en œuvre d'une attaque de type empoisonnement du cache DNS (« *cache poisoning* »), trouvée par Dan Kaminsky, sur le serveur de nom de domaine Bind ;
- émission, via une pièce jointe d'un courriel, d'un fichier PDF exploitant la vulnérabilité JBIG2 présente dans certaines versions du logiciel Adobe Reader (CVE-2009-0658[2]) ;
- mise en place de différentes attaques sur des applications Web :
  - SPIP, récupération d'une sauvegarde complète de la base du site sans authentification (BID:36008[3]) ;
  - PhpMyAdmin, mise en place d'une porte dérobée par injection de code dans le fichier de configuration principal (CVE-2009-1151[4]).

Le script Bro suivant permet de détecter efficacement l'exploitation de la faille de Dan Kaminsky.

Script de politique Bro :

```
@load alarm
@load dns

module DNS;

const KAMINSKY_DETECTION_LEVEL = 100 &redef;

redef enum Notice += { Kaminsky, };
global kaminsky_reported: set[addr] &write_expire = 1 min;
global kaminsky_t: table[addr] of table[string]
```

[1] Outil disponible à l'adresse http://ha.ckers.org/slowloris/.
[2] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-0658.
[3] http://www.securityfocus.com/bid/36008/info.
[4] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-1151.

```
                    of count &write_expire = 1 min;

event dns_request(c: connection, msg: dns_msg, query: string,
                  qtype: count, qclass: count)
{
   local d_ip = c$id$resp_h;

   # Si la cnx n'est pas en UDP, Kaminsky ne s'applique pas.
   if(!is_udp_port(c$id$orig_p))
         return;

   # La requête doit être de type A.
   if(qtype != 1)
         return;

   # La faille ne s'applique que si la récursivité est demandée.
   if(!msg$RD)
         return;

   kaminsky_t[d_ip] = table() &mergeable;
   kaminsky_t[d_ip][query] = 0;
}

event dns_A_reply(c: connection, msg: dns_msg, ans: dns_answer, a:
addr) {
   local d_ip = c$id$resp_h;
   local q = ans$query;

   if(d_ip in kaminsky_reported)
         return;

   # Si la cnx n'est pas en UDP, Kaminsky ne s'applique pas.
   if(!is_udp_port(c$id$orig_p))
         return;

   # La faille ne s'applique que si la récursivité est demandée.
   if(!msg$RD)
         return;

   if(d_ip !in kaminsky_t)
         return;

   if(q !in kaminsky_t[d_ip])
         return;

   ++(kaminsky_t[d_ip][q]);

   local cnt = kaminsky_t[d_ip][q];

   if(cnt >= KAMINSKY_DETECTION_LEVEL) {
         NOTICE([$note=Kaminsky,
                 $conn=c,
                 $msg=fmt("Attempt to inject poison record (Kaminsky)
(%s replies to \"%s\" query (on %s)", cnt, q, d_ip)
                ]);
         add kaminsky_reported[d_ip];
   }
```

Figure 4 – Script Bro permettant de détecter l'attaque de Kaminsky

Le script suivant détecte avec succès une attaque s'appuyant sur Slowloris. Il est en outre capable de différencier une attaque réalisée avec cet outil d'un utilisateur effectuant un grand nombre de requêtes (utilisation d'un « aspirateur » de site Web par exemple).

Script de politique Bro :

```
@load signatures
@load alarm
@load http

const SLOWLORIS_DETECTION_LEVEL = 100 &redef;

redef enum Notice += { SlowLoris, };
global report_once = T &redef;
global slowloris_reported: set[addr] &persistent;
global slowloris_t: table[addr] of table[addr]
                    of set[conn_id] &write_expire = 30 sec;

# Handler appelé lors de l'émission d'une nouvelle requête HTTP. Lève
# une alerte "SlowLoris" au delà de 100 connexions *actives* depuis
la
# même adresse source et vers le même serveur destination.
event http_request(c: connection, method: string, original_URI:
string,
                   unescaped_URI: string, version: string)
{
   local d_ip = c$id$resp_h;
   local s_ip = c$id$orig_h;
   local id = c$id;

   if(s_ip in slowloris_reported)
         return;

   if(d_ip !in slowloris_t)
         slowloris_t[d_ip] = table() &mergeable;

   if(s_ip !in slowloris_t[d_ip]) {
         slowloris_t[d_ip][s_ip] = set() &mergeable;
   }

   if(id !in slowloris_t[d_ip][s_ip]) {
         add slowloris_t[d_ip][s_ip][id];
         local l = length(slowloris_t[d_ip][s_ip]);

         if(l >= SLOWLORIS_DETECTION_LEVEL) {
               NOTICE([$note=SlowLoris,
                       $conn=c,
                       $msg=fmt("Slowloris (%s -> %s => %s cnx)",
                                s_ip, d_ip, l)
                      ]);

               if(report_once == T)
                     add slowloris_reported[s_ip];
         }
   }
}

# Handler appelé lors de la fin d'une connexion HTTP. Il s'agit dans
# ce cas de supprimer la connexion de la liste des connexions HTTP
# actives.
event http_message_done(c: connection, is_orig: bool,
                        stat: http_message_stat)
```

```
{
   local d_ip = c$id$resp_h;
   local s_ip = c$id$orig_h;
   local id = c$id;

   if(d_ip !in slowloris_t)
         return;

   if(s_ip !in slowloris_t[d_ip])
         return;

   if(id in slowloris_t[d_ip][s_ip])
         delete slowloris_t[d_ip][s_ip][id];
}
```

Figure 5 – Script Bro permettant de détecter l'utilisation de Slowloris

La signature suivante a permis de détecter l'émission d'un fichier PDF, contenant la charge active permettant d'exploiter la vulnérabilité d'Adobe Reader sur JBIG2 :

```
signature smtp_containing_jbig_1 {
   ip-proto == tcp
   dst-port == 25
   tcp-state established,originator
   event "SMTP Suspicious JBIG2 pdf file sent from email "
   payload /.*JVBERi0x.*/
   payload /.*[A-Za-z0-9_\x2f][EUk0]pCSUcyRGVjb2Rl/
}
```

Figure 6 – Signature permettant de détecter l'envoi d'un courriel infecté par la faille JBIG2

Cependant, un attaquant peut contourner ces signatures en envoyant le fichier, par exemple, dans une archive compressée. Afin de se prémunir de ces possibilités de contournement, il est nécessaire de disposer d'un analyseur particulier permettant d'analyser les pièces jointes dans les transactions SMTP, de les décompresser (comme cela peut-être fait par l'analyseur HTTP) puis de l'envoyer au mécanisme de signatures.

Les signatures suivantes permettent de détecter efficacement l'attaque sur SPIP, tout en limitant le taux de faux positifs :

```
signature spip_cve-2009-3041 {
   ip-proto == tcp
   tcp-state established,originator
   event "SPIP vulnerability detected : CVE-2009-3041"
   http /.*ecrire\//
   http /.*(\?|&)exec=install/
   http /.*(\?|&)reinstall=non/
   http /.*(\?|&)transformer_xml=/
   http /.*(\?|&)nom_sauvegarde=/
   requires-reverse-signature http-200
}

signature http-200 {
   ip-proto == tcp
   tcp-state responder
   payload /^HTTP\/[0-9\.]+ 200/
}
```

Figure 7 – Signatures permettant la détection de l'attaque sur SPIP (BID:36008)

Afin d'éviter qu'un attaquant puisse contourner cette signature en changeant l'ordre des paramètres de l'URL, la signature a été décomposée. Par ailleurs, une alerte ne sera levée que si le serveur répond sans erreur (code 200) grâce au mécanisme de dépendance de signatures (« `require-reverse-signature` »). Ceci permet de ne pas générer de faux positifs lorsque la version de SPIP n'est pas vulnérable.

Résistance de Bro aux techniques d'évasions

Fragmentation TCP/IP

Bro est robuste vis-à-vis des techniques d'évasion d'IDS simples, qui reposent sur la fragmentation IP et TCP. En revanche, il est possible pour un attaquant d'exploiter une ambiguïté de la spécification du protocole IP en ce qui concerne la réémission de séquence avec recouvrement. En pratique, différents algorithmes ont été implémentées dans la gestion de la pile TCP/IP. Bro s'appuie sur l'algorithme du système d'exploitation BSD, qui est différent de celui mis en œuvre par le système Linux, utilisé par les serveurs de la plateforme de tests. Ceci permet à un attaquant de masquer une attaque ou de générer des faux positifs.
En outre, Bro est sensible à l'insertion de certains paquets malformés (qui sont rejetés par les serveurs) et qui permettent potentiellement de masquer des attaques.

L'envoi d'un paquet « RST » invalide permet de mettre à mal tous les analyseurs protocolaires situés au dessus de TCP. Il est ainsi possible de contourner le mécanisme de recherche de motifs (et donc le système de détection dynamique de protocole), mais aussi les analyseurs protocolaires comme HTTP ou SMTP par exemple.

Cependant, cette faiblesse est à pondérer par le fait que de tels paquets « RST » sont généralement bloqués par les pare-feux. Ceux-ci font effectivement du suivi de connexion (« *connection tracking* ») et sont donc en mesure d'intercepter de tels paquets aux numéros de séquences ou d'acquittement incohérents.

Cette limitation peut s'avérer problématique dès lors qu'il s'agit de détecter des attaques internes, sur un réseau local. C'est par exemple l'objet du script `blaster.bro`, qui vise à identifier les machines Windows infectées par le virus Blaster.

Evasion HTTP

Bro est partiellement robuste quant à sa manière de décoder le protocole HTTP.

Il est capable de déjouer les techniques de contournement simples de `libwhisker` (mauvais codage des caractères, requêtes malformées, etc.).

Dans la mesure où Bro ne fait pas de normalisation des URL avant de les envoyer au mécanisme de recherche de motifs, il est vulnérable aux attaques par offuscation d'URL. L'impact de telles faiblesses est cependant pondéré par le fait que les signatures reposent peu fréquemment sur la recherche de chemins complets dans les URL.

Il n'est pas possible de mettre en œuvre de techniques d'évasions basées sur une mauvaise interprétation du format UTF-8. En revanche, Bro ne permet pas d'identifier des motifs accentués dans ce format .

Efficacité du traitement des signatures

Les données suivantes ont été mesurées :

- l'occupation mémoire de Bro à l'issue de l'étape de chargement des signatures ;
- l'occupation mémoire de Bro à l'issue de la génération de trafic du jeu de test, c'est-à-dire lorsque le DFA a été généré et lorsque la charge est redescendue à son niveau moyen ;
- le nombre de paquets rejetés.

Le mécanisme de gestion des signatures de Bro est relativement robuste. L'occupation mémoire est limitée, y compris lorsque la taille de la base de signatures est importante.

Gestion des défaillances

Bro dispose de mécanismes rudimentaires de gestion des défaillances rudimentaires. Certains mécanismes sont expérimentaux (adaptation du niveau de décodage suivant la charge). La fonctionnalité de sérialisation[1]/désérialisation semble à première vue intéressante (elle peut être utilisée pour reconfigurer Bro en minimisant la perte de paquets). La fonctionnalité de « chien de garde » s'est avérée efficace. Toutefois, l'utilisation de ce type de mécanisme n'est pas sans danger. De plus, la fonctionnalité de redémarrage automatique de Bro n'a pu être testée.

2.3.4.2 Avis d'expert sur la résistance des mécanismes

L'efficacité et la robustesse des mécanismes de Bro sont variables :

- Bro possède une bonne capacité de détection dès lors que les scripts et les signatures nécessaires à la détection des attaques sont correctement spécifiés. Bro offre un niveau d'expressivité élevé, notamment grâce au langage de script Bro et à l'intégration du mécanisme de signatures avec le mécanisme de gestion d'événements (possibilité de prendre en compte le contexte, capacité à discerner le succès d'une attaque, possibilité de compléter la détection par comparaison de signatures avec des aspects comportementaux, etc.). Toutefois, la base de signatures fournie avec Bro, importée en 2004 de la base de Snort et proposée à titre d'exemple, n'est plus d'actualité. Elle n'a pas permis de détecter les attaques récentes qui ont été mises en œuvre sur la plate-forme de test. Elle ne doit donc bien être considérée qu'à titre d'exemple.
- La résistance des analyseurs protocolaires (en particulier TCP) n'est pas suffisante pour contrer un attaquant mettant en œuvre des techniques d'évasions simples et connues.
- Le mécanisme de gestion des signatures est relativement robuste. L'analyse du code source et des articles publiés par les développeurs montre que les problématiques liées à la complexité des algorithmes d'évaluation des expressions régulières ont été prises en compte dès la conception du mécanisme. La consommation mémoire reste dans des limites acceptables (y compris lorsque la taille de la base de signatures est importante). Bro semble relativement robuste en ce qui concerne le passage à l'échelle jusqu'à 10 000 signatures (au-delà, la consommation mémoire augmente linéairement).

[1] Enregistrement de variables de script dans la mémoire de masse.

2.3.5 Analyse des vulnérabilités (conception, implémentation…)

2.3.5.1 Liste des vulnérabilités connues

Aucune vulnérabilité publique n'est recensée sur cette version du produit Bro.

2.3.5.2 Liste des vulnérabilités découvertes lors de l'évaluation et avis d'expert

L'analyseur protocolaire de Bro n'implémente pas l'intégralité des vérifications imposées par la RFC de TCP. Il est notamment possible d'émettre un paquet *reset* qui est considéré comme invalide par les piles TCP/IP des machines attaquées mais qui provoque l'arrêt du suivi de la connexion par Bro. Il est également possible de leurrer l'IDS en utilisant certaines techniques implémentées par les outils *Fragroute* et `libwhisker` Ceci permet à un attaquant de masquer les attaques qu'il réalise sur le réseau surveillé par Bro.

Les faiblesses du moteur d'analyse protocolaire TCP/IP de Bro conduisent à des vulnérabilités qui sont, en théorie, facilement exploitables par un attaquant et qui permettent de mettre à mal tous les mécanismes d'analyse de Bro. Toutefois, ce type d'attaques implique la génération de trafic TCP/IP mal formé. Ce type de trafic est généralement filtré au préalable par différents éléments du réseau (routeurs, firewalls, etc.). Ceci limite l'impact de ce type de vulnérabilités. Il est cependant difficile d'évaluer a priori le niveau de protection qu'offrent les différents équipements du réseau. En effet, ce niveau dépend des éléments utilisés, de leur configuration, de la localisation de l'attaquant dans le réseau, etc. L'attaquant peut être interne au réseau surveillé, par exemple s'il s'agit d'un logiciel malveillant. De plus, la documentation et la cible de sécurité de Bro n'imposent aucune restriction sur l'utilisation de Bro. Elles devraient préciser la nécessité d'utiliser un mécanisme de filtrage et de normalisation des paquets réseau en aval de la sonde Bro.

Les faiblesses du moteur d'analyse protocolaire HTTP de Bro conduisent à des vulnérabilités facilement exploitables par un attaquant. Toutefois, ces vulnérabilités ne concernent essentiellement que le moteur de reconnaissance des signatures : un attaquant peut leurrer les signatures de type HTTP qui reposent sur la reconnaissance d'un motif d'URL faisant intervenir plusieurs niveaux de répertoire (par exemple `/.*\/test/index.html/`). Il est alors possible d'effectuer des requêtes dont la sémantique est équivalente à l'URL de la signature, et qui seront interprétées de manière identique par les serveurs, mais que Bro ne reconnaîtra pas (par exemple `/test/./index.html`). Il s'agit essentiellement d'une limitation fonctionnelle de Bro (qui ne décode pas complètement les URL) dont les utilisateurs de Bro doivent être conscients lorsqu'ils développent des signatures.

2.3.6 Analyse de la facilité d'emploi et préconisations

2.3.6.1 Cas où la sécurité est remise en cause

Les lacunes de la documentation de Bro peuvent mener à des cas d'utilisation où la sécurité est remise en cause.

En effet, la documentation de Bro est globalement incomplète et obsolète, et il est parfois difficile de connaitre le rôle exact de certains événements émis par les analyseurs ou de certaines variables internes. Ceci peut aboutir à l'écriture de scripts ayant un comportement différent de celui attendu.

Par ailleurs, le comportement par défaut de certains mécanismes s'avère être relativement peu intuitif, comme c'est le cas pour le mécanisme de gestion de signatures. Ces points sont détaillés ci-dessous.

Lacunes dans la documentation

La documentation sur les analyseurs n'est ni exhaustive ni à jour. Par conséquent, il est souvent nécessaire de deviner le sens des événements et des variables, ce qui nécessite parfois une lecture du code source. C'est par exemple le cas de l'événement « `connection_EOF` » qui est généré lorsqu'une des parties ferme la connexion TCP.

Par ailleurs, la signification réelle des événements émis est parfois non conforme à ce qui est annoncé dans la documentation. Ainsi, l'événement « `connection_pending` » n'est censé être levé que lorsque Bro se termine, alors que différents cas de tests ont montré le contraire. C'est aussi le cas de l'événement « `new_connection` » qui est documenté comme étant un événement lié à TCP alors que celui-ci est émis lors de la création de chaque nouvelle connexion au sens de Bro.

Mécanisme de gestion des signatures

Par défaut, le mécanisme de reconnaissance de signatures ne vérifie celles-ci que sur le début du flux réassemblé (premiers paquets de la connexion). L'utilisateur n'est donc pas nécessairement conscient de cette limitation et la base de signatures activée peut être contournable lorsque la charge du flux réassemblé est de taille importante. Ce comportement est configurable grâce à la variable « `dpd_match_only_beginning` », au nom peu explicite et qui est très mal documentée.

Enfin, l'utilisation de la recherche de motifs dans des messages HTTP (signature utilisant le mot-clé `http`) peut être source d'ambigüités. Par exemple, il n'est pas possible de réaliser une signature sur le protocole HTTP permettant de rechercher un motif dans les en-têtes.

Environnement de déploiement de Bro

La documentation n'indique aucunement que l'IDS Bro doit être déployé au sein d'un réseau où le trafic est préalablement filtré ou normalisé afin d'éviter toute ambigüité. Ceci est pourtant nécessaire pour se protéger contre les attaques évoquées au paragraphe 2.3.4.1, par exemple l'attaque de type « *RST-packet injection* ». L'utilisateur doit être pleinement conscient des risques auquel il s'expose en cas d'utilisation de Bro dans un environnement dont le trafic n'est pas normalisé ou filtré.

2.3.6.2 Recommandations pour une utilisation sûre du produit

Afin de se protéger contre les attaques de type injection de paquet RST (« *RST-packet injection* »), il est nécessaire qu'un prétraitement soit opéré sur le trafic réseau. Il est ainsi possible de mettre en place du filtrage à l'aide d'un pare-feu rejetant tous les paquets jugés non conformes par le mécanisme de suivi de protocole. Il pourra en outre être nécessaire de mettre en place un système de normalisation du trafic TCP/IP afin de supprimer les ambiguïtés intrinsèques à l'utilisation de ce protocole.

Ce problème doit être pris en compte lors de la mise en place de l'IDS Bro dans le cadre de la détection d'intrusion réalisée sur un réseau local (propagation de vers ou de virus, etc.). Ce même problème se pose de la même manière si l'IDS est placé volontairement en amont du pare-feu, dans l'objectif d'observer, par exemple, toute les tentatives d'attaques.

Comme cela a été montré dans l'analyse de la robustesse du protocole HTTP, Bro ne réalise pas de normalisation de l'adresse demandée dans la requête. Cette limite doit être prise en compte lors de l'écriture de scripts ou de signatures. Il est ainsi possible de contourner le problème en décomposant la recherche de motifs. Bro deviendra donc insensible à la présence des séquences de contournement incriminées. En revanche, cette configuration rendra la sonde plus sensible aux faux positifs.

Lors de l'activation du mécanisme de signature avec un objectif plus large que la simple détection dynamique de protocole (DPD), il peut être utile de désactiver la variable Bro « `dpd_match_only_beginning` ». Si la valeur de cette variable est positionnée à « `T` », alors Bro ne vérifiera que le premier kilo-octet de données de chaque connexion. Il sera nécessaire de le faire même si la détection de protocole est inutilisée. La documentation officielle (répertoire « `docs/` » ou Wiki) ne fait pas état de l'existence de cette variable au nom peu explicite.

2.3.6.3 Avis d'expert sur la facilité d'emploi

Bro est un outil dédié à des utilisateurs experts. La qualité inégale de sa documentation ne facilite pas son utilisation. Il est parfois nécessaire d'analyser le code source ou des scripts existants afin d'assimiler le fonctionnement de Bro. Certains points ambigus ou non documentés peuvent conduire à une mauvaise utilisation du logiciel.

2.3.7 Accès aux développeurs

Le code source complet est disponible depuis le site de Bro (http://bro-ids.org). L'évaluateur n'a pas eu de contact avec la communauté de développeurs de Bro durant l'évaluation.

2.4 Analyse de la résistance des mécanismes cryptographiques

Le produit évalué ne comporte pas de mécanismes cryptographiques.

3 La certification

3.1 Conclusion

L'évaluation a été conduite conformément aux règles en vigueur, avec la compétence et l'impartialité requise pour un centre d'évaluation agréé.

Ce certificat atteste que le produit Bro soumis à l'évaluation répond aux caractéristiques de sécurité spécifiées dans sa cible de sécurité [ST], aux limites près indiquées dans le présent rapport.

3.2 Restrictions d'usage

Ce certificat porte sur le produit spécifié au chapitre 1.2 du présent rapport de certification.

L'utilisateur du produit certifié devra s'assurer du respect des objectifs de sécurité sur l'environnement d'exploitation spécifiés dans la cible de sécurité [ST], suivre les recommandations énoncées dans le présent rapport de certification au paragraphe 2.3.6.2 ainsi que celles se trouvant dans les guides fournis [GUIDES] avec le produit.

Annexe 1. Références documentaires du produit évalué

[CDS]	SGD005-CDS-1.02 du 17 août 2009 Disponible sur www.ssi.gouv.fr/site_rubrique54.html
[RTE]	SGD005-RTE-1.1 du 02 novembre 2009
[Guide]	SGD005-DOC-1.00 du 03 août 2009 Disponible sur www.ssi.gouv.fr/site_article80.html
[DOC1]	Manuel de l'utilisateur (« *User Manual* ») Disponible sur www.bro-ids.org
[DOC2]	Manuel de référence (« *Reference Manual* ») Disponible sur www.bro-ids.org
[DOC3]	Fichier « *CHANGES* » Disponible sur www.bro-ids.org

Annexe 2. Références liées à la certification

Décret 2002-535 du 18 avril 2002 relatif à l'évaluation et à la certification de la sécurité offerte par les produits et les systèmes des technologies de l'information.	
[CER/P/01]	Procédure CSPN-CER/P/01 Certification de la sécurité offerte par les produits et les systèmes des technologies de l'information, DCSSI, disponible sur www.ssi.gouv.fr/site_article80.html
[CSPN]	Certification de sécurité de premier niveau (CSPN) des technologies de l'information, version 2.4, phase expérimentale, n° 915/SGDN/DCSSI/SDR/CCN du 25 avril 2008. Critères pour l'évaluation de sécurité de premier niveau des technologies de l'information, phase expérimentale, version 1.4. Méthodologie d'évaluation en vue de la CSPN et contenu attendu du RTE, phase expérimentale, version 1.3. Documents disponibles sur www.ssi.gouv.fr/site_article80.html

Annexe 3. Glossaire

Acronyme	Définition
BPF	*BSD Packet Filter* (filtre de paquets BSD).
BSD	*Berkeley Software Distribution* (système d'exploitation de type Unix développé par l'université de Berkeley).
DAG	*Data Acquisition Generation* (cartes d'acquisition réseau développées par la marque Endace).
DFA	*Deterministic finite-state machine* (automate à états finis déterministe).
DMZ	*Demilitarized zone* (zone démilitarisée).
DPD	*Dynamic Protocol Detection (mécanisme de Bro permettant de détecter automatiquement et dynamiquement les protocoles)*
FAQ	*Frequently asked questions* (foire aux questions).
FTP	*File Transfer Protocol* (protocole de transfert de fichiers).
GPG	*GNU Privacy Guard* (logiciel de chiffrement/signature de courriels).
HIDS	*Host Intrusion Detection System* (IDS système).
HTTP	*Hypertext Transfer Protocol* (protocole de transfert hypertexte).
ICMP	*Internet Control Message Protocol* (protocole Internet utilisé pour véhiculer des messages de contrôle et d'erreur).
IDMEF	*Intrusion Detection Message Exchange Format* (format de message pour l'échange entre systèmes de détection d'intrusion, standard défini par la RFC 4765).
IDS	*Intrusion Detection System* (système de détection d'intrusions).
IP	*Internet Protocol* (protocole Internet).
NFA	*Nondeterministic finite state machine* (automate à états finis non deterministe).
NIDS	*Network Intrusion Detection System* (IDS réseau).
OS	*Operating System* (système d'exploitation).
RFC	*Requests for comments* (document officiel décrivant les aspects techniques d'Internet).
SMTP	*Simple Mail Transport Protocol* (protocole de transfert de courriels).
SSH	*Secure Shell*.
SSL	*Secure Socket Layer* (protocole de sécurisation des échanges sur Internet).
TCP	*Transmission Control Protocol* (protocole de contrôle de transmission).
UDP	*User Datagram Protocol* (protocole de datagramme utilisateur).
URI	*Uniform Resource Identifier* (identifiant uniforme de ressource permettant d'adresser une ressource physique ou abstraite sur un réseau).
URL	*Uniform Resource Locator* (localisateur uniforme de ressource permettant d'adresser les ressources du *World Wide Web*).