IETFにおける侵入検知情報交換方式の問題点と対策

金岡 晃[†] 岡本栄司[††]

侵入検知システム（IDS）の情報共有のためのメッセージ形式とメッセージ交換プロトコルの標準化が進んでおり，現在それぞれIDMEF，IDXPとしてIETFのワーキンググループIDWGから提案されている．そこでインターネットドラフトとして公開されている双方を実装し評価を行ったところ，パフォーマンスに関して有用性があることは分かったが，DoS攻撃を受ける可能性があること，および，なりすましによる攻撃の可能性があることが判明した．本研究ではこれらの危険性を指摘し，対策を行うことでこれらの問題点を解消できることを示した．

The Problems and Countermeasures of Intrusion Detectin Exchange Format in IETF

Akira Kanaoka[†] and Eiji Okamoto[††]

"Intrusion Detection Message Exchange Format (IDMEF)" and "Intrusion Detection Exchange Protocol (IDXP)" have been proposed by IDWG as the standards for data formats and exchange procedure between IDSs. We implemented these formats and protocols and evaluated them. Though its performance is high, we found the possibility of DoS attack and the possibility of "man in the middle" attack. In this paper, we introduce these possibilities and propose new methods against these vulnerabilities.

1. はじめに

現代のネットワークはその急激な発展により社会に多大な恩恵をもたらしているが，同時に大きなセキュリティの問題もかかえている．セキュリティ上の問題を解決する技術あるいはシステムは数多く存在し，IDS（Intrusion Detection System：侵入検知システム）もそのシステムの1つである．

現在IDSは商用でも多数のシステムや手法が存在し，それぞれに長所や短所を持っている．使用する側は，1つのIDSでなくいくつかの種類のIDSを利用してそれらから得られる情報を見ることで，総合的にセキュリティを確保している．

このような利用が増える中，増加する商用IDSのデータ形式はベンダによりさまざまで利用者にとってけっして利用しやすい状況ではなくなっている．さらに，最近では分散配置されたIDSを用いてその情報の相関をとることでさらに高度な侵入検知を目的とするものもでてきている．

このため，IDSが発する情報表示形式や交換方法を標準化しようと動きが現れ，現在IETFのワーキンググループによってそのインターネットドラフトが提出されている．

我々はそれら公開されているドラフトより実装を行い評価を行った[1)]．その結果これら標準化の有用性を示すことができたが，同時にDoS攻撃の可能性が存在すること，および，なりすまし攻撃の可能性があることが判明した．本論文ではこれらの問題を示し，その対策を提案する．

2章ではIETFにおける標準化を紹介し，3章ではその標準化においてかかえる問題点について述べる．4章でそれらの問題点を解消する対策を提案し，シミュレーションによりその有効性を示す．5章でその問題点の対策に関する考察を行い，将来の課題を示した．

2. IETFにおける侵入検知標準化動向

IDSに関する標準化はIETFにおけるワーキンググループIDWG（Intrusion Detection Working Group）[2)]によって進められており，その成果が現在インターネットドラフトとして提出されている．IDWG

† 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

では，以下のものを作成している．

- Intrusion Detection Message Exchange Requirements（侵入検知メッセージ交換要件，以下 IDMER）[3]
- Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language（XML）Document Type Definition（侵入検知メッセージ交換フォーマットデータモデルと XML DTD，以下 IDMEF DTD）[4]
- The TUNNEL Profile（TUNNEL プロファイル）[5]
- The Intrusion Detection Exchange Protocol（侵入検知交換プロトコル，IDXP）[6]

IDMER では，疑わしいイベントを報告するアナライザとそれを受けるマネージャという役割を仮定し，そこでの警告のフォーマットと通信方法を IDMEF によって標準化するとしている．

IDMER ではさらにメッセージフォーマットや内容に対する要件や，交換プロトコル（Intrusion Detection Communication Protocol，IDP）に対する要件，さらには交換されるメッセージと交換プロトコルは独立して考えることが述べられている．

2.1 IDMEF

IDMEF（Intrusion Detection Message Exchange Format，侵入検知メッセージ交換フォーマット）は XML で記述される．IDMEF DTD では IDMER でのメッセージフォーマットの要件を満たすための XML での表現方法，文書型定義（DTD）などが定義されている．しかし，実際には IDMEF で表現したいものが DTD では正確に表現できないため，将来的には XML 内容の定義が DTD ではなく XML Schema へと変更される予定である．

この IDMEF メッセージには 2 種類，Alert と Heartbeat があり，Alert の UML による表現を**図 1** に，Heartbeat の UML による表現を**図 2** に示す．また，Alert の DTD による表現をあわせて次に示す．DTD が UML の汎化（Generalization）を表現できないために，UML で汎化として表現されているクラス群が DTD では子要素として表現されていることが分かる．

```
<!ELEMENT Alert          (
  Analyzer, CreateTime, DetectTime?,
  AnalyzerTime?, Source*, Target*,
  Classification+, Assessment?,
  (ToolAlert | OverflowAlert |
  CorrelationAlert)?, AdditionalData*
)>
```

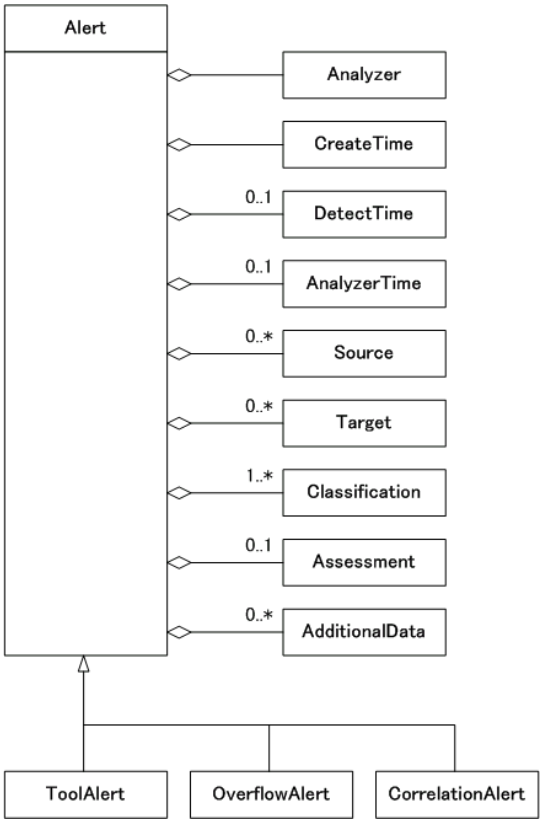

図 1 Alert の UML 表現
Fig. 1 UML expression of Alert class.

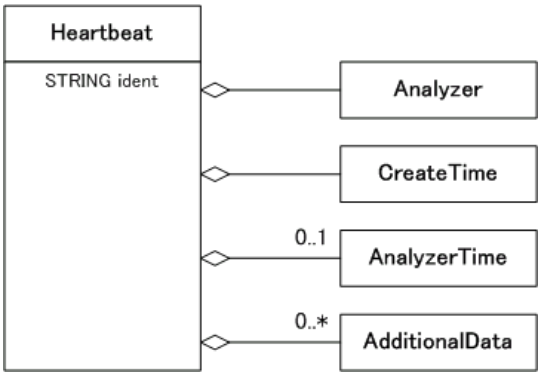

図 2 Heartbeat の UML 表現
Fig. 2 UML expression of Heartbeat class.

2.2 IDXP

IDMER によるメッセージ交換プロトコル（IDP）の要件として，

- 信頼あるメッセージ伝達
- ファイアウォールとの相互作用
- 相互認証
- メッセージの機密性
- メッセージの完全性

- 情報源ごとの認証
- サービス妨害対策（推奨）
- メッセージ重複防止（推奨）

があげられている．

IDXP はこれらの条件を満たす実装の 1 つとして提供される．IDXP は単体でプロトコルをなすものではなく，BEEP（Blocks Extensible Exchange Protocol）というアプリケーションプロトコルフレームワーク上のプロファイルとして実装される．上記の要件の 1 つに「ファイアウォールとの相互作用」とあるが，これはファイアウォールを挟んでの侵入検知情報交換を行うにあたってもそのセキュリティのレベルを落とすことのない通信を可能にしなければならない，としている．これは IDXP と同じく IDWG で提案されている TUNNEL プロファイル[5]で実現され，その他の要件は BEEP 上での TLS（Transport Layer Socket）において暗号スイート TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA を利用することで実現される．この暗号スイートは，鍵交換に用いる公開鍵は DSS 署名付きのその場限りの DH 鍵，暗号化に CBC モードのトリプル DES，ハッシュには SHA-1 を利用する．

2.3 標準化の有用性

侵入検知情報を交換するための形式と交換方法を標準化するということは，現在さまざまな種類が存在する IDS において異機種間での情報交換が可能になるという利点が存在する．文献 1) においては，ログを検査するタイプの IDS とファイルの完全性を検査するタイプの IDS という，まったくタイプの異なる IDS の協力により実際上素早い対応が可能であることが示されている．

3. Heartbeat の問題点

3.1 Heartbeat とは

Heartbeat は，アナライザからマネージャへと定期的に送られる IDMEF メッセージで，アナライザの生存確認やネットワークの正常さを伝えるために用いられる．

文献 4) によると，IDMEF メッセージを送信する側のアナライザが Heartbeat を使用することは任意であるが，メッセージを受信する側のマネージャは Heartbeat への対応が必須となっている．

3.2 DoS 攻撃の可能性

IDXP 上で IDMEF メッセージ（Alert，Heartbeat）を送る流れは**図 3** に示すようになっているが，実際にメッセージを送る場合にはいくつかのパターンが考えられる．

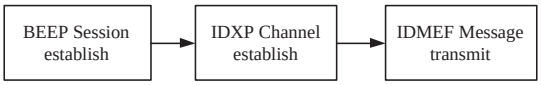

図 3 IDMEF メッセージ送信処理の流れ
Fig. 3 Basic flow of IDMEF message transmission.

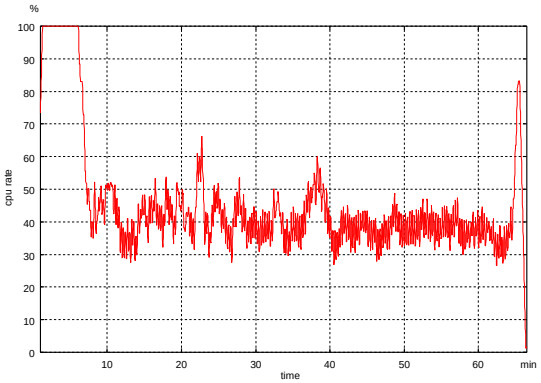

図 4 DoS 攻撃を受けているマネージャの CPU 使用率状況
Fig. 4 CPU usage of manager under DoS attack.

(1) セッション接続→チャンネル接続→メッセージ送信
(2) チャンネル接続→メッセージ送信（セッションはつねに張られている状態）
(3) メッセージ送信（セッション，チャンネルはつねに張られている状態）

現実での利用は，IDMEF メッセージを送る頻度やマネージャがどれくらいのアナライザをかかえているかによって大きく変わってくる．

本論文は IDWG が提案する方式において攻撃可能性があることを示し，それらへの対処方法を述べるものである．そのためにその攻撃の範囲を IDMEF と IDXP に限定し，BEEP への大量要求や TLS への大量要求に関する問題は本論文では考慮しない．

我々は発表[1]で，上記 (2) のパターンで大量のチャンネル接続要求と Heartbeat 送信を行うことでマネージャへの DoS 攻撃が可能であることを示したが，実際には上記いずれのパターンでも DoS 攻撃が可能である．

図 4 は (2) のパターンにおいてマネージャへの DoS 攻撃シミュレーションを行ったときのマネージャ側 PC の 1 分間での平均 CPU 使用率を表したグラフである．このシミュレーションではより軽量なマネージャでも DoS 攻撃か可能であるかを調査するために TLS を利用せずに実験を行った．攻撃開始から 5 分を過ぎたところから CPU の平均使用率の低下が始まり，その後 50%を切る状態で推移していることが分かる．**図 5** は 1 分間での平均 CPU 使用率が 50%を切っている状態

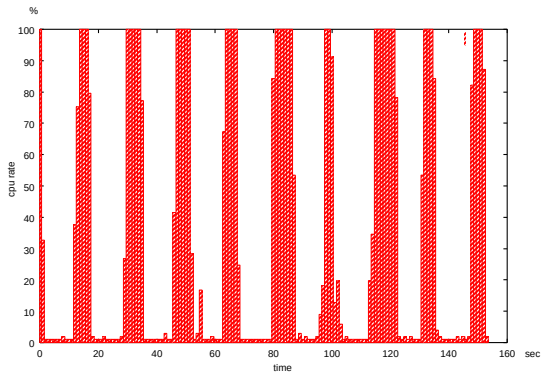

図 5 ある時間帯におけるマネージャの CPU 使用率
Fig. 5 CPU usage of manager in certain time.

の時間帯における 1 秒間ごとの CPU 使用率のグラフを切り出したものである．

このグラフから，1 分間平均が 50%を切っているのが，実は CPU 使用率が 100%と 0%を交互に繰り返している結果であるということが分かる．この状況下では断続にサービスがダウンしていることが示されている．

また，この時間帯に攻撃とは関係のない第三者が IDMEF メッセージの送信を試みるという実験もあわせて行った．その際に第三者は上記 (3) のパターンの接続を選択した．その結果，アナライザは BEEP セッションの接続要求を出すが，タイムアウトが発生し接続が不可能であった．このことから事実上 DoS 攻撃が成功していることが分かる．この実験においては第三者は攻撃者とは違うパターンでの接続をしているが，このようなパターンの混在は実環境においてはアナライザごとのメッセージ送信頻度の違いにより十分に考えられ，それらを考慮して実験を行った．

3.3 IDS 無効化の可能性

Heartbeat によって生存確認を行うという考え方は，アナライザが何らかの原因で攻撃者によって乗っ取りをうけた場合に DoS 攻撃の可能性があるだけでなく，なりすましをされてしまうことによってマネージャに気づかれることなく他への攻撃を実現する IDS 無効化の可能性もかかえている．

IDMEF と IDXP を実装する IDS（アナライザ）を考えたとき，そのシステムは実際に侵入検知による EOI（Event Of Interest）に従い Alert を発生させる部分と，定期的に Heartbeat を送信する部分に分かれる（**図 6**）．アナライザがイベント検出部と Heartbeat 送出部に分かれて実現されているならば，攻撃側はイベント検出部だけを強制終了させ，イベント検出を無効にする一方 Heartbeat 送出部は変わらず動作させ続けることでマネージャ側に気づかれることなくアナライザを無効化させることが可能である．また，イベント検出部と Heartbeat 送出部が 1 つのシステムとして実現されている場合，あるいは分割していても強固な連携を持っているシステムの場合であっても，アナライザのシステムをすべて強制終了させて攻撃者自身が乗っ取り前と変わらぬタイミングで Heartbeat を送信し続けることによって，マネージャが気づくことなくアナライザを無効化してしまうことが可能になる．このことにより，アナライザが検知対象としている範囲においてマネージャ側へメッセージ通知されることなく攻撃が実行されてしまう．

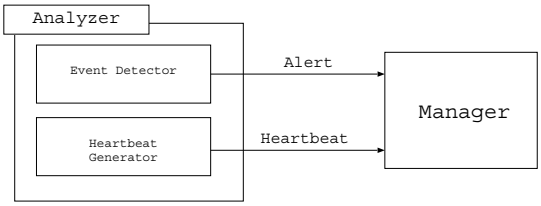

図 6 イベント検出部と Heartbeat 送出部
Fig. 6 Event detector and Heartbeat generator in analyzer.

DoS 攻撃は開始することでマネージャ側に対して送信元アナライザになんらかの問題が起きていることが確認できてしまうという反面，なりすましによる攻撃はマネージャにはまったく気づかれないという特徴を持つ．

3.4 攻撃の実現可能性

上で示した DoS 攻撃あるいは IDS 無効化の攻撃は，アナライザの乗っ取りが最初に必要になる．その後，乗っ取ったアナライザを利用してマネージャへの DoS 攻撃や，アナライザの監視対象への攻撃が可能になる．これらは実際には TLS を使用していれば，暗号化や認証が必要であるためにその実現は簡単ではない．

アナライザ自身が乗っ取られる状況としては，具体的にはアナライザシステムの脆弱性をつくことなどがあげられる．むしろ TLS を信用しすぎることにより，攻撃を受けていることがマネージャに伝わらない，あるいはマネージャが攻撃に気づくまでに時間がかかることもありうる．

4. 対策技術

3 章で述べた攻撃は，アナライザが乗っ取りを受けることで起きる．マネージャ側がそれらの攻撃に対して耐性を持つためには，以下の 2 点が考えられる．

(1) Heartbeat 以外の手段を利用してアナライザの生存確認を行う．

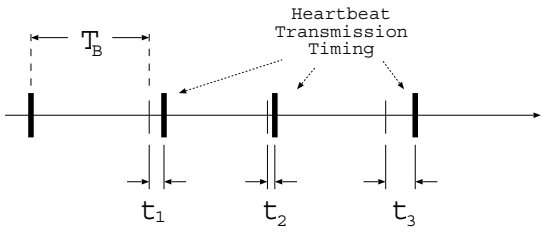

図 7 THH の原理
Fig. 7 Rational of Time Hopping Heartbeat (THH).

(2) Heartbeat に細工を加えることにより，Heartbeat の偽造を見破る仕組みを提供する．

(1) を利用するということは，Heartbeat の存在自身の否定にもなりうる．しかし仕様においてはマネージャ側の Heartbeat 対応は必須となっているので現段階においては得策ではない．また，別の方法によって生存確認を行うことはアナライザとマネージャに余分な負担（通信量や処理量）をかけることになる．

本研究では (2) の方を選択し，その実現方法として THH（Time Hopping Heartbeat），また THH の利用が困難である場合も想定して乱数を直接データに書き込む方式を提案し，これらの提案方式の処理に関しての実験も行った．

4.1 Time Hopping Heartbeat

文献 4) によると Heartbeat はアナライザからマネージャへと定期的に送られるメッセージであり，例として 10 分ごとや 1 時間ごとの送信をあげているが，厳密に定期的に送らなくてはならないという規定はない．そこで本研究では Heartbeat を送る間隔に変化を加えることによりなりすましを防ぐ．

ここで，Heartbeat を送信する基本間隔 T_B とし，毎回の送信間隔を変化量 t_i とする．このとき毎回の送信間隔 T_i を以下のように表す．

$$T_i = T_B + t_i. \quad (1)$$

この t_i の系列を毎回変えることで，Heartbeat の送信間隔に変化をつけることができる（**図 7**）．この系列はアナライザとマネージャ間でしか生成できない必要があり，また，第三者が今までのこの系列を盗聴していても次の値に何がくるかが分からないようにしなくてはならない．そのために，この系列には暗号用乱数を元に値を生成する．

暗号用乱数生成の同期をとるためにアナライザとマネージャは乱数発生の種（Seed）を共有しなければならないが，これはアナライザ起動時にアナライザが生成しマネージャへ自動的に送信するという形をとる．種の送信時には通信路は暗号化されている必要がある．

IDXP を用いた通信では，図 3 にあるように IDMEF メッセージを送るにあたり BEEP のセッションを張る必要があり，その後 IDXP のチャンネルを開く．TLS を用いた通信を利用する場合には IDXP のチャンネルを開く前に，TLS を用いてセッション自体を暗号化している．そこで，このセッション暗号化のやりとりの後，IDXP チャンネルを開く前に BEEP の Option タグを利用してアナライザは暗号乱数発生の種を送信する．

アナライザは種を送信した後，その種自体は破棄する．また，乗っ取り時のなりすましを防ぐためには種を変える際（再起動する際）にマネージャがその変更を把握しておく必要がある．さらにできれば，種の変更に関する情報をマネージャが正確に把握するためにアナライザの停止や再起動などはマネージャ主導で行うことが望ましい．再起動の場合，アナライザが種をあらためて生成してマネージャに送信する．

アナライザ側では，この系列に従って Heartbeat を送るタイミングを操作して Heartbeat を送信する．

受け取るマネージャ側では，受け取った時刻を確認し，あらかじめ準備してある Heartbeat の到着予定時刻と比較し，その Heartbeat が正しく送信されているかどうかを検証する．

時刻の確認は，Heartbeat のメッセージに必ずある CreateTime タグを利用する．次に Heartbeat の DTD を示す．

```
<!ELEMENT Heartbeat  (
    Analyzer,
    CreateTime,
    AnalyzerTime?,
    AdditionalData*
)>
<!ATTLIST Heartbeat
   ident CDATA '0'
>
```

THH を利用することで，攻撃者によるアナライザの乗っ取りが発生した場合でも，THH を偽造することは暗号用乱数系列を正確に発生させなければならないことから IDS 無効化の攻撃は困難であることが分かる．また DoS 攻撃に関しても，THH の送信間隔に異常が発生した場合にそれ以降のメッセージをすべて遮断することで DoS 耐性を持つことが可能である．さらに，THH は Heartbeat に必ず必要とされる CreateTime タグの情報を使うことにより，送信するデータの量を増やすことなく乗っ取り対策を行うことが可能である．

しかし，現実的な実装を考えた場合 THH には注意

すべき点がある．Heartbeat の基本間隔 T_B が長い場合であれば問題なく THH を使用することができるが，T_B が短い場合には変化量系列 t_i の幅をどのように定義するかが難しくなる．まず，変化量の幅は T_B と比較して小さいことが望ましい．なぜなら，基本間隔 T_B と変化量 t_i の幅がほぼ等しいことは，ほぼランダムな間隔で Heartbeat を送ることを意味し，生成された乱数によっては連続して Heartbeat 送信が発生することがあるからである．マネージャは複数のアナライザを持つことが十分に考えられるために，このような連続した Heartbeat 発生はマネージャの処理を一時的に増大させる危険性もあるために，避けるべきである．

IDMEF における日付・時刻の情報はミリ秒の単位までの表現が可能であるので，表示に関しての問題は発生しない．しかし，実際にミリ秒までの単位を正確にあわせて Heartbeat を生成することは非常に難しいと考えられる．

そこで THH での実現が難しい場合も考えて，THH を利用せずに乱数データを直接 Heartbeat に付加する方法もあわせて提案する．

4.2 乱数データの付加

Heartbeat は，データとして Analyzer と CreateTime を必ず持たなければならないが，追加の情報として AdditionalData というものが利用可能である．この AdditionalData の部分は，内容の記述が自由にできる場所でこの部分を利用することで IDMEF ではベンダ特有の情報送信を認めている．

THH の利用がむずかしい場合には，生成した乱数を直接この AdditionalData の部分に書き込むことで THH と同じ効果を持たせることが可能である．

しかし，THH とは異なり付加情報として乱数を送信するので，本来の Heartbeat より送信サイズが増えるという点があり，THH が利用できる状況であれば THH を利用することが望ましい．

4.3 処理時間の比較

マネージャは多数のアナライザから情報を受け取ることが考えられ，担当するアナライザの数が多ければ，マネージャの負担も大きくなる．提案した 2 手法がどれほどの負担をマネージャに与えるかを調査するために，マネージャの処理時間の比較実験を行った．IDMEF に従ったメッセージを生成するアナライザとそれを受け取るマネージャを用意し，IDXP 上で Heartbeat の送信を行う．IDXP では IDMEF メッセージ送信が完了した後に，受け取ったマネージャは ok タグあるいは error タグのメッセージを送らなくてはならない．実験では，受信確認後すぐに Heartbeat を送信することを 1,000 回繰り返し，1,000 回の Heartbeat 送信にかかった時間を測定した．

表 1 使用した PC のスペック
Table 1 Spec. of used hardware.

	CPU	Memory
マネージャ	PentiumIII 1.26 GHz	1,024 MB
アナライザ	PentiumIII 1.13 GHz (×2)	1,024 MB

実験は 4 種類行った．以下にその種類をあげる．

(1) アナライザ：通常 Heartbeat 送信
マネージャ：受信のみ
(2) アナライザ：通常 Heartbeat 送信
マネージャ：受信後，XML をメモリにロード（アナライザ，日付確認）
(3) アナライザ：対策 Heartbeat 送信（THH）
マネージャ：受信後，XML をメモリにロード（アナライザ確認，THH を考慮した日付確認）
(4) アナライザ：対策 Heartbeat 送信（乱数付加）
マネージャ：受信後，XML をメモリにロード（アナライザ，日付，乱数確認）

実験に利用したプログラムの言語には Java を選択した．BEEP の部分は beepcore[7]が提供している beepcore-java を使用した．また IDMEF メッセージと IDXP については自作し，乱数発生については Java の Security パッケージにある SecureRandom クラスを利用した．乱数は SHA-1 を用いて発生される擬似乱数アルゴリズムを使用した．

シミュレーションはマネージャ側での処理に要する時間の比較を目的にしているが，THH は本来基本間隔と乱数によって生成される変化量によって生成タイミングが決まるので，アナライザは連続して送信するようには設計されていない．そこで THH を用いた場合にはシミュレーション用に THH の連続生成を可能にするよう変更を加えて実験を行った．

マネージャ側についても，通常のマネージャであれば日付情報に問題がある場合には即座に対象アナライザからのメッセージ受付を拒否するが，シミュレーション用に偽造判定の機構を停止したシステムへと変更した．

THH についてはアナライザとマネージャにこれらの変更を加えることによって処理自体の時間計測を行った．

また実験に使用したハードウェアのスペックを**表 1** に示す．

(1) の場合は，XML ドキュメント自体をメモリにロードしていない，つまり Heartbeat を受け取っただけで生存確認としている場合を想定したものである．

表 2 Heartbeat 処理時間の比較
Table 2 Comparison of Heartbeat processing time.

	1 回あたりの平均所要時間 (msec)
(1)	75.628
(2)	79.814
(3)	80.454
(4)	79.838

表 2 の結果をみると，この場合の処理が一番軽いことが分かるが，実際に XML をメモリへロードして，その内容確認を行った (2) でも 1 回あたりの処理時間の差の平均が 4 msec 程度であり実際には処理に大きな差がないことが分かる．また，(3) の THH や (4) の乱数付加といったなりすましに対策を施した Heartbeat の処理に対して処理を行わない (2) の処理時間と比較すると，その差はそれぞれ 0.640 msec，0.024 msec とほとんど差がないことが分かり，対策処理がけっして重いものではないことが分かる．

5. 考察・課題

5.1 考　　察

本研究では，IETF における侵入検知情報共有のための標準化である IDMEF と IDXP に関して，生存確認のために定期的にアナライザからマネージャに送られる Heartbeat についてマネージャに対する DoS 攻撃の可能性が存在すること，さらには侵入検知システムが無効化されてしまう可能性が存在することが判明したことについて触れた．

IDMEF や IDXP の仕様では，DoS 耐性があることが推奨され，またなりすましに対する耐性が必須となっている．確かに TLS を利用することで大幅に耐性が上がるが，実装を考えた場合にアナライザの役割をする実体（プログラムやシステム自体）にはセキュリティホールなどの脆弱性がある可能性は否めない．それら脆弱性を利用することでアナライザが一度乗っ取られてしまうと TLS による DoS 耐性やなりすまし耐性は簡単に崩壊してしまう．

そこで本研究ではそのような乗っ取りへの対策技術としてアナライザとマネージャが第三者には生成できない系列の発生を利用して送信タイミングを変化させる Time Hopping Heartbeat（THH）を提案した．さらに，THH が利用できない状況も考えて直接系列をデータに書き込む方法もあわせて提案した．それら 2 つの提案に関して実際に実験を行い，その有効性を示した．

THH，乱数付加ともにパフォーマンスとしてほぼ同等の水準をみせたが，THH は通常の Heartbeat と送信データサイズが変わらないことや，THH に対応していないマネージャなどでも通常に受け取ることが可能であることなどから，間隔変化量の制限が許す範囲であれば THH を利用するほうが有効であると考えられる．

5.2 課　　題

Heartbeat は侵入検知情報の交換要件には入っていない類の情報であり，メッセージの仕様でアナライザの生存確認のために現れてきたものである．しかし，Heartbeat を利用することによって得る利益より，Heartbeat が存在する場合の危険性の方が大きいと我々は考える．

現在の仕様ではマネージャ側の Heartbeat 対応が必須となっているが，通常の使用では危険性があることより，本論文で示した対策技術を利用することを IDWG へ提案することを考えている．

参 考 文 献

1) 金岡　晃，岡本栄司：IDS 標準化：実装と評価，コンピュータセキュリティシンポジウム 2002 論文集，pp.449–454 (2002).
2) Intrusion Detection Exchange Format (idwg). http://www.ietf.org/html.charters/idwg-charter.html
3) Wood, M. and Erlinger, M.: Intrusion Detection Message Exchange Requirements, Internet-Draft (Aug. 2002).
4) Curry, D. and Debar, H.: Intrusioin Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition, Internet-Draft (June 2002).
5) New, D.: The TUNNEL Profile, Internet-Draft (Dec. 2002).
6) Feinstein, B., Matthews, G. and White, J.: The Intrusion Detection Exchange Protocol (IDXP), Internet-Draft (June 2002).
7) beepcore.org. http:// www.beepcore.org
8) SILICON DEFENCE. http://www.silicondefense.com
9) Kothali, P.: Intrusion Detection Interoperability and Standardization, SANS Reading Room: Intrusion Detection.

付　　録

A.1 Heartbeat メッセージの例

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE IDMEF-Message PUBLIC "-//IET
F//DTD RFC XXXX IDMEF v1.0//EN""idmef-
```

```
message.dtd">
<IDMEF-Message version="1.0">
  <Heartbeat ident='5963'>
    <Analyzer analyzerid='lcis-analyze
r001' >
      <Node ident='0' category='dns' >
        <Address ident='0' category='i
pv4-addr' >
          <address>192.168.1.2</addres
s>
        </Address>
      </Node>
    </Analyzer>
    <CreateTime ntpstamp='0xc18ba797.0
x77300000' >
      2002-11-25T04:31:35+09:00
    </CreateTime>
  </Heartbeat>
</IDMEF-Message>
```

(平成 14 年 11 月 29 日受付)
(平成 15 年 6 月 3 日採録)

金岡　晃

1975 年生．1998 年東邦大学理学部情報科学科卒業．2001 年東邦大学大学院理学研究科情報科学専攻修士課程修了．現在，筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士課程に在学中．

岡本　栄司（正会員）

1950 年生．1973 年東京工業大学工学部電子工学科卒業．1978 年東京工業大学大学院理工学研究科電子工学専攻博士課程修了．工学博士．同年日本電気（株）中央研究所入社．その後，北陸先端科学技術大学院大学情報科学研究科，ウィスコンシン大学，東邦大学をへて 2002 年より筑波大学電子・情報工学系教授，現在に至る．1993 年〜1994 年文部省在外研究にてテキサス A&M 大学数学科客員教授．グラフ理論，通信理論，数理計画，アルゴリズム，情報セキュリティをはじめとする情報数理工学の教育・研究に従事．1990 年電子通信学会論文賞，1993 年情報処理学会ベストオーサ賞受賞．著書「暗号理論入門」（共立出版），「電子マネー」（岩波書店）等．IEEE シニア会員．ACM，電子情報通信学会，情報理論とその応用学会，応用数理学会，日本セキュリティ・マネージメント学会，IACR（International Association for Cryptologic Research）会員，IJIS（International Journal of Information Security）編集長．