Linuxホスト運用の課題と展望

荒木靖宏

Hewlett-Packard Labs Japan

日本Linux協会(JLA)

サービスポリシーとセキュリティポリシー

目的
- システムの目的とセキュリティ保持の目的を定める

原則
- サービスとセキュリティのバランス決定

方針
- 原則実現の基本方針。責任を明確にする

手順
- 具体的な手順や方法を定める

サービスへのあくなき要求

- いつでも使えること
  - ⇒High Availability
- 十分な速度で利用できること
  - ⇒Scalability
- ⇒ 現在のトレンドは、具体的に数値を設定:SLAへ
  - 使えないときは通知されること
  - 数値目標の達成を掲示
  - 運用に加えて監視の重要性大

一次被害

- 攻撃の対象は多岐にわたる
  - デーモンそのもの、デーモンが呼びだすサブプログラム、認証機構、全く別のプログラム

対策
- 安全なプログラムの使用に尽きる
- 外に晒すインターフェースの制限がよくある保護法
  - よく考えられたアクセス制御がよくある方法

単一Linuxホスト内でのセキュリティの確保

ユーザ&グループベースのセキュリティに依存

可能な努力
- デーモン毎に動作ユーザ/グループを別にする
- 必要なユーザ以外からファイルを見えないようにする
- setuid,setgidの排除

努力の例

sendmail vs postfix, qmail

apache vs thttpd

アクセス制御(from network)

カーネルレベルでのIPの制御
- ipchains, iptable

- 評価対象:
  - source, destination, インターフェース
- 処理:
  - ACCEPT, DENY, REJECT, MASQ(IP masquerade), REDIRECT, RETURN

アプリケーションレベル
- それぞれのアプリケーション(当然)
- tcpwrapper+inetd、tcpserver、xinetd
  - 広く使われているスーパーデーモン(application layer)
  - 柔軟なアクセス制御を提供

二次被害

- 一次攻撃は防げない場合がある
- 影響範囲の制限が必要

対策と現実

- 二次以降のセキュリティ対策を考えるとできうる限りの権限分離が必要

- 現実
  - サービス毎にサーバ分離できない
  - ハードウェアが無い
  - 分離方法がわからない
  - その場限りの対応の積み重ねで分離不可能に

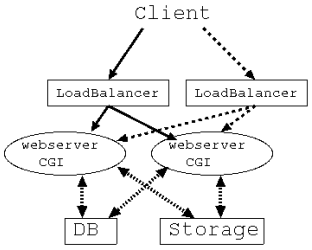
サービスの切り分け

システム設計の見直し
- 人間による見直しが確実
- 論理的に分解

toolの利用
- lsof
  - 動作中のプロセスが使用しているファイル情報を示す
- pstree, ps, ldd..

大規模特定サービスモデル

webをつかったサービスなどでは必然的に仕事が分別される



- 相関がきちんと定義されて、影響は小
- side effectとしてsecurity向上も計れる
- 十分な分離が必要でそれが許される希少例

Trusted Linux

カーネルレベルでのリソースのチェックと分離

コンパートメント毎に独立に動作する

リソースのコンパートメント化
- ラベリング
  - プロセス、シェアードメモリ、セマフォ、ネットワーク
- chroot
  - Filesystem

Source/Destinationによるアクセス定義が可能
- コンパートメント、ホスト、ネットワークを指定

SELinux

カーネルサブシステムとして実装

SID(Security ID)
- システム全てのfileについて定義(regexp使用可能)
  - user:role:domain
  - user:role:type
  - 例:
    - /var/tmp(/.*) system_u:object_r:tmp_t
    - /etc/rc.d/rc.sysinit system_u:object_r:initrc_exec_t

API提供(libsecure)
- アプリケーションに手をいれる必要あり

複数ホストによる大規模システム構築

問題多数

- 仕事(役割)の分担
- 個々のホストのsetup
- システムの監視の問題

ディストリビューションを使わない選択

ディストリビューションは必要悪という立場

- Linuxシステムの差異を産む
- アプリケーション等インストール情報の喪失懸念
-

一方で。。

- インストールは?
- 本当にそのインストール情報、わかっていますか?
- 大量に同じ目的のシステムを作れますか?

ディストリビューションが必要ない例

- 十分な知識がある
- 専門化したシステムに用いる
- 複製が必要ない場合

ディストリビューション

利点

- そもそも多数のコピーを柔軟に作製するのに便利なシステム
- Linuxシステム構築の現実的な方法

沢山あるけど何を選ぶ? 選択基準

- 一般的な見方
  - パッケージの充実度
  - 使い易さ
- サーバを構築するなら
  - バグ、セキュリティホールのupdate速度
  - 商用プログラムを使うならその対応
  - Trinuxのような最小限のディストリビューションを選ぶ手も

ディストリビューション利用の指針

- 自分が把握しているパッケージのみ
- 必要最小限
- 最新のパッケージ
- 必要な付加プログラムはパッケージングする
  - 同一のテスト環境の構築も容易

障害発見方法

- 人が連絡
  - 連絡窓口
    - 管理者/サービス提供元への電話、FAX、メール
  - rfc2142で求められるアドレス
    - abuse, security, noc, postmaster, hostmaster, usenet, news, webmaster, www, uucp, ftp
- 監視ツール
  - 省力化
  - ミスの減少
  - 通常気がつかないミスの発見

システムの監視

最終的には監視あってはじめてわかる

システムとしての監視

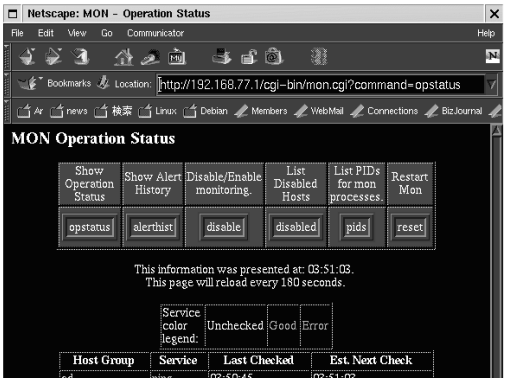
- サービス内容を満たしているかどうか
- 動作テスト

構成ホスト個々の監視

- file構成、内容
  - tripwire, ViperDB

監視ツール / mon

構成:スケジューラ + 監視プログラム + 通知プログラム
それぞれ独立,容易な拡張



mon / 監視:

- マルチレベルでの監視が可能:
  - 附属プログラム:
    - L7: SMTP, Telnet, FTP, NNTP, POP-3, IMAP, LDAP, DNS, mSQL, MySQL, RPC
    - L4: 任意tcp port
    - L3: ホスト間RTT値, ping
    - upime, process,Disk残容量
- 必要なら、監視以外のツールももちろん利用可能

mon / 通知 + スケジューラ

- 週単位での時間帯制御
- 異常検知後のプログラム指定

設定例

```
hostgroup router cisco7504    ⇒router group 作成
watch router                  ⇒router group の監視を宣言
  service ping                ⇒pingサービスについて
  interval 5m                 ⇒5分ごとの監視
    monitor fping.monitor     ⇒fping.monitorプログラムで監視
    period wd {Mon-Fri} hr {7am-10pm}
      alert mail.alert ar@domain.com
```

Heartbeat

機器間がお互い生きているかを調べる

- Linuxでは多くの場合で利用される標準技術
  - ⇒他へも多く応用される
- いくつかの方法を提供
- 必要な情報をメッセージにして広告

Heartbeatのメディア

- Ether
  - Shared-Ether
    - 信頼性:良 レイテンシ:可 スケーラビリティ: 良 コスト:安
  - 1vs1 Ether
    - 信頼性:良 レイテンシ:良 スケーラビリティ: 良 コスト:高
  - Ether Multicast
    - 信頼性:良 レイテンシ:可 スケーラビリティ: 優 コスト:高
- シリアルポート
  - 信頼性:優 レイテンシ:良 スケーラビリティ: 可 コスト:安
- IrDA
  - 信頼性:良 レイテンシ:良 スケーラビリティ: 良 コスト:安

Heartbeatのメッセージ

- フォーマット:簡単な文字情報
  - >>>
  - name=value
  - <<<
- ノード間通信:
  - メッセージタイプ、ソースノード、ノート時刻、シーケンス番号、ロードアベレージ、TTL、メッセージ到達時刻
  - ディスティネーションノード(Optional)
    - 再送依頼(firstseq, lastseq)

Heartbeatのメッセージ(2)

- 集められるステータス情報:
  - st/status:
    - dead or unknown
  - info/reason: (Optional)

- IPアドレス関係:
  - ip-request:他ノードにテイクオーバーを依頼する
    - ipaddr:
  - ip-request-resp:ip-requestに対する返事
    - ipaddr:
    - ok:

So much for today!

どうもありがとうございました。

Happy vacationing!