Webアプリケーションの
セキュリティ限界に迫る

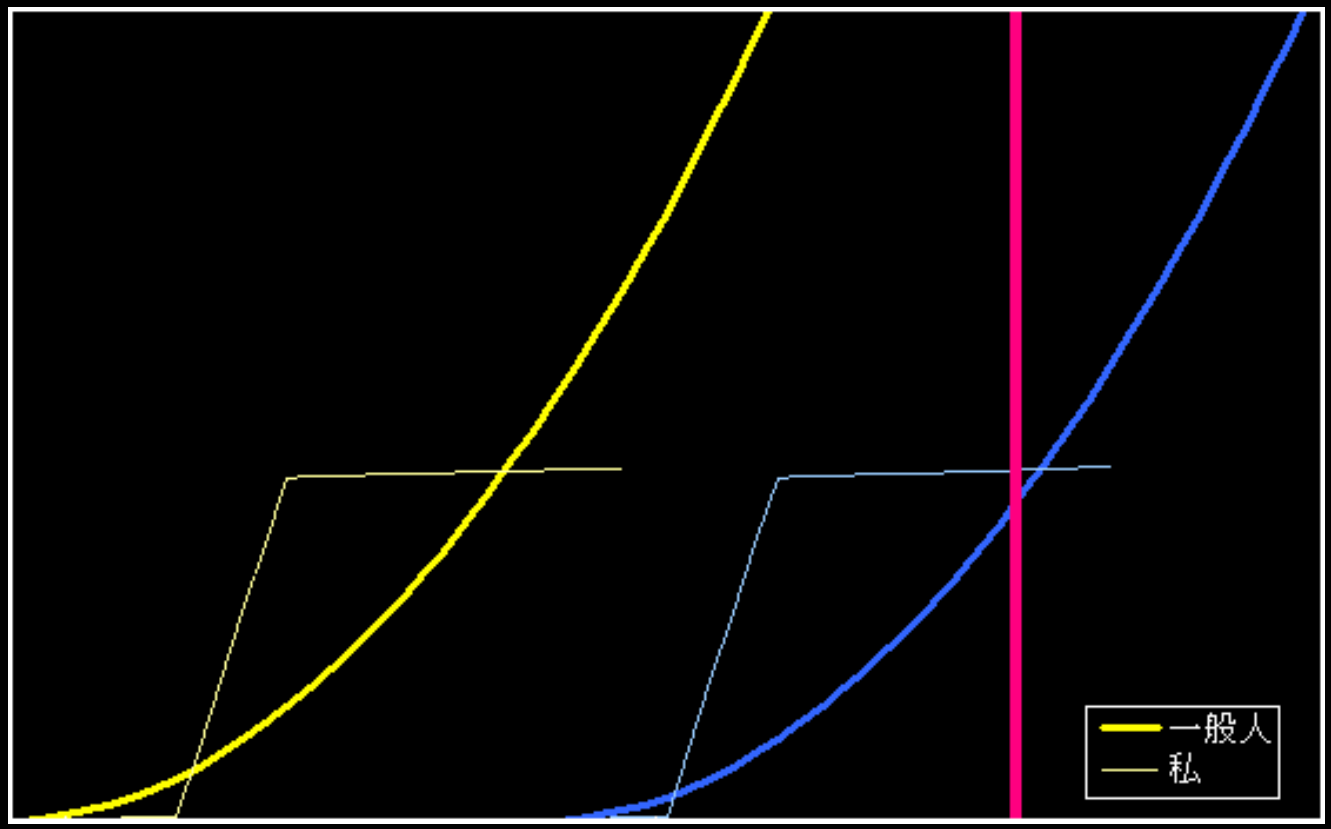
国分裕
bun@devnull.jp

この資料に関して

- 本資料の内容を不正アクセスに使用しないでください。
- 本資料を使用した結果起こる問題に関して、著者及びNT-Committee2は一切責任を負いかねます。

自己紹介

- 某社コンサルタント(セキュリティ)
- "週"記 (http://www.devnull.jp/tdiary/)



セキュリティを確保するには

- それぞれのフェーズで対策をする
  - 構築
  - 検査
  - 防御
  - 検知

成熟してるとは言い難い

どうすればよいのか

- 何ができて、何ができないかを知る
- できることは確実に
- できないことは諦める　(;´Д`)

Webの推移(1)

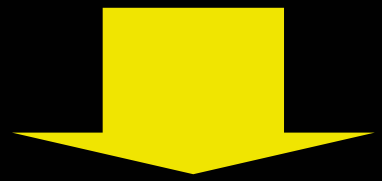
- 静的コンテンツの特徴
  - 毎回同じコンテンツ
  - 誰がアクセスしても同じコンテンツ
  - ディレクトリ単位のアクセス制御

Webの推移(2)

- 動的コンテンツへの要望
  - アクセスするたびに違うコンテンツを表示したい
  - アクセスする人毎に違うコンテンツを表示したい
  - もっと細かいアクセス制御をしたい

Webの推移(3)

- 静的なコンテンツの転送
  - 同一内容のコンテンツ
  - すべてのコンテンツを公開
  - Webサーバによる認証



- 動的なコンテンツの転送
  - クライアント毎に異なるコンテンツ
  - 部分的なコンテンツの公開
  - Webアプリケーションによる認証

Webアプリケーションとは

- クライアントとの通信にHTTPを使うアプリケーション
- 構成要素
  - HTTPD(Apache, IIS...)
  - アプリケーションサーバ(Tomcat, WebLogic...)
  - データベース(Oracle, MySQL, SQL Server...)
  - 個別アプリケーション
    - Java, Perl, ASP, PHP, Ruby...

個別アプリケーションの問題

- インターネット上に公開される
- 短い開発期間
- 限られた予算

- セキュリティのスキルの欠如/意識がない人が作る場合も…

Webアプリケーション潜む脆弱性

- Verbose Error Messages
- HTML Comments
- Known Directory
- Known CGI File
- Configuration File Disclosure
- Backup File Disclosure
- SQL Injection
- Cross-Site Scripting
- Buffer Overflow
- OS Command Injection
- Meta Character Injection
- Directory Traversal
- Null Injection
- User-Agent Manipulation
- Referrer Manipulation
- Debug Commands
- Extension Manipulation
- Frame Spoofing
- Brute/Reverse Force
- Session Hi-Jacking
- Session Replay
- Session Forging
- Password Recovery

SQL Injection(1)

- アプリケーションで使っているSQL文を操作
- アプリケーションの想定外のSQL文を実行

検索条件の変更
データの参照
データを追加･変更･削除

- SQL (Structured Query Language)

SQL Injection(2)

http://www.example.com/login.asp?
user=foo&pass=bar

SELECT * FROM USERS WHERE
name='foo' AND password='bar'

SQL Injection(3)

http://www.example.com/login.asp?
user=admin&pass=bar'or''%3D'

SELECT * FROM USERS WHERE
name='admin' AND password='bar'or''='

adminユーザでログイン

SQL Injection(4)

http://www.example.com/login.asp?
user=&pass=''; DELETE FROM USERS --

SELECT * FROM USERS WHERE
name='' AND
password=''; DELETE FROM USERS --'

すべてのデータを削除

Directory Traversal(1)

- アプリケーションで使用するファイル名を操作

任意のファイルの読み出し
任意のファイルの追加･変更･削除

Directory Traversal(2)

http://www.example.com/cgi-bin/news.cgi?
file=0123

```
open(FILE, '/var/www/news/0123.txt');
while(<FILE>){print;}
close(FILE);
```

Directory Traversal(3)

http://www.example.com/cgi-bin/news.cgi?
file=../cgi-bin/news.cgi%00

```
open(FILE, '/var/www/news/../cgi-bin/news.cgi');
while(<FILE>){print;}
close(FILE);
```

ソースコードの漏洩

Directory Traversal(4)

http://www.example.com/cgi-bin/news.cgi?
file=../../../etc/passwd%00

```
open(FILE, '/var/www/news/../../../etc/passwd');
while(<FILE>){print;}
close(FILE);
```

構築編

- セキュアなアプリケーション構築が基本

プログラミング上の留意点

その前に

- 設計上の留意点

設計上の留意点

- 入出力データの定義
  - 文字種と形式
  - 長さ
  - どこからどこにデータがわたるか
  - データの意味
  - 可変か固定か
  - HTTPSで保護すべきデータかどうか
- 画面の遷移

プログラミング上の留意点

- 入力値チェック
  - 設計に従い正しい値が入力されているか確認
- サニタイジング
  - 有害文字(列)を無害化

入力値チェック

- 設計に従い正しい値が入力されているか確認する。
- 正規表現を使うのが便利
  - 郵便番号欄

    → ¥d{3}-¥d{4}
  - セッションID

    → [0-9a-f]{32}

サニタイジング(1)

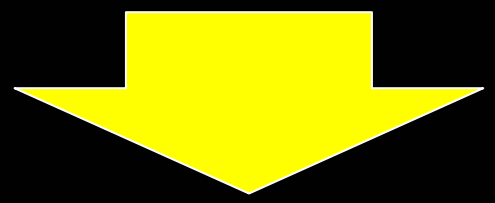
- 入力チェックを潜り抜けた文字を出力する場合に、それぞれの出力形式での特殊文字を無効化する

サニタイジング(2)

- CSVの場合
  - 「, 」が含まれている
    → 全体を " " で括る
  - 「"」が含まれている
    → 全体を " " で括る
    → 「"」を「""」に変換する
  - 改行が含まれている
    → 全体を " " で括る

サニタイジング(3)

- HTMLを出力する場合
- SQLを出力する場合
- OSコマンドを出力する場合



仕様書読んで考えてください

構築編まとめ

- 設計をしっかりする
- 入力値チェックとサニタイジング
- バグ・ミスをできるだけ排除する

検査編

- 構築上のミスを見つける
  - 設計書レビュー
  - コードレビュー
  - ブラックボックステスト

検査手法(1)

- 設計書と見比べながら、不正な値を入力
  - 設計書: 数字4桁

    ↓

    アルファベット・4桁以上の数字・2バイト文字

    -1、+1

  - 設計書: 全角カタカナ

    ↓

    ひらがな・半角カタカナ

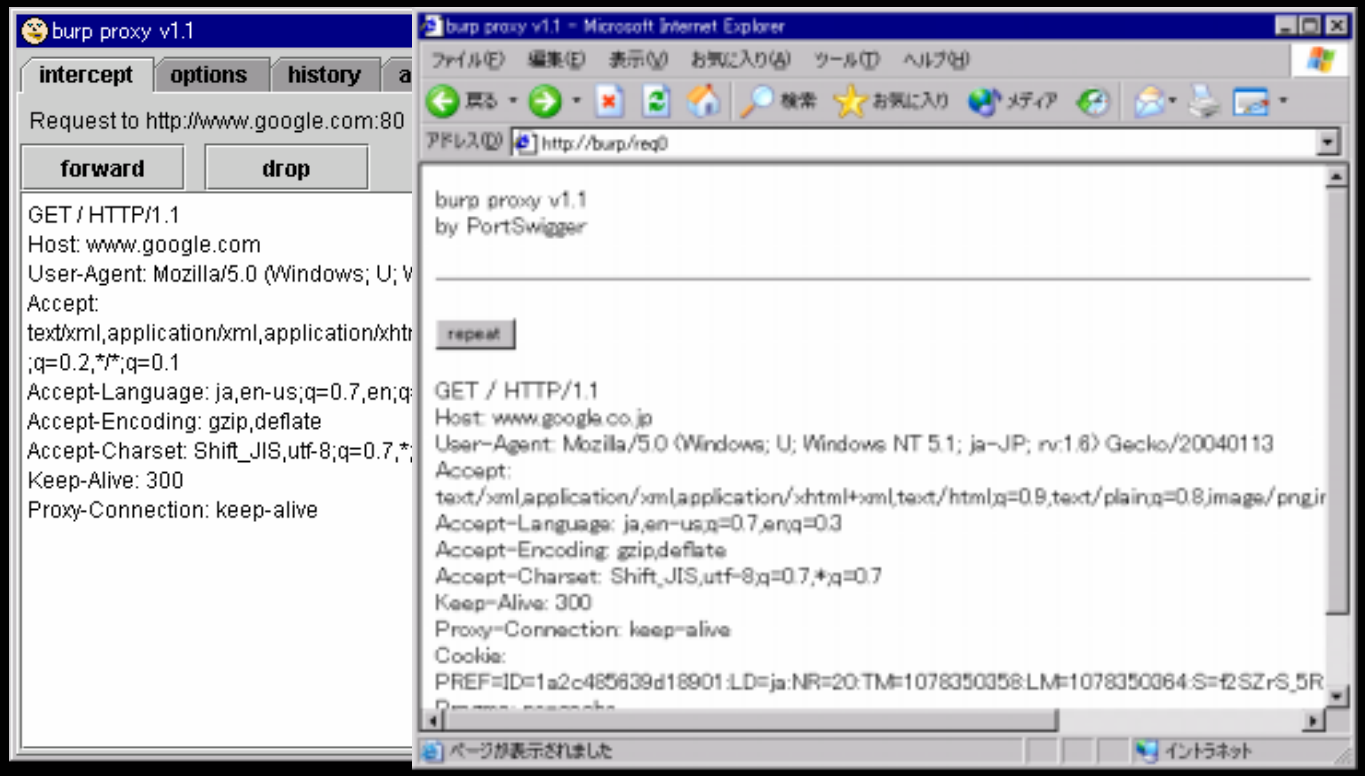
検査手法(2)

- **特殊文字を入力**
  - !”#$%&’()-^Y=~|@[]{}‘;:+*,.<>?_
  - **おすすめは「’」(シングルクオーテーション)**
  - %00、%0D、%0A

検査を楽にするツール

- Burp proxy

http://portswigger.net/proxy/



私が検査中に考えること

- どのくらいのスキルレベルの人かなぁ
- どんなプログラムが動いてるのかなぁ
- ここはこういう動きを期待してるんだろうなぁ
- この人はこの辺ミスしそうだなぁ

システムごとに作ってる人が違うと

- つなぎ目に問題ないかなぁ

など

自動検査ツール

- AppScan (Sanctum)
  - http://www.sanctuminc.com/
- ScanDo (KaVaDo)
  - http://www.kavado.com/
- WebInspect (SPI Dynamics)
  - http://www.spidynamics.com/
- WebProbe (ソフテック)
  - http://www.softek.co.jp/Sec/WebProbe/

自動検査 vs 手動検査(1)

- 自動検査
  - 早い
  - 結果が操作者に依存しない
  - 検査漏れ・検査不可なアプリケーションもある
- 手動検査
  - 複数の情報からさらに詳細な検査
  - ブラウザでアクセスできる範囲はどこでも
  - 手間・時間がかかる
  - 検査者のスキルレベルによる差

検査編まとめ

- 早い段階で検査を
- 検査ツールを過信しない
- 自動検査と手動検査の住み分けを
  - 自動検査：大規模に定量的な検査
  - 手動検査：重要なサイトに絞って
  - 併用

防御編

- 人間はミスをする
- システムで守れるところは守る

アプリケーションファイアウォール

- mod_security
- guard3
- Guardian@JUMPERZ.NET
- AppShield

mod_security(1)

- **Intrusion Detection And Prevention**
- Apache **のモジュール**

http://www.modsecurity.org/

mod_security(2)

```
LoadModule security_module modules/mod_security.so
<IfModule mod_security.c>
    SecFilterEngine On
    SecAuditEngine On
    SecAuditLog logs/audit_log
    SecFilterScanPOST On
    SecFilterDefaultAction "pass,log"
</IfModule>
```

- Snort→mod_security

http://www.modsecurity.org/documentation/converted-snort-rules.html

guard3(1)

- IIS**に組み込む**ISAPI **フィルタ**
- **パターンマッチによるフィルタリングルール**

http://www.trusnet.com/tools/guard3/index.html

guard3(2)

- HKLM¥SOFTWARE¥CCS¥guard3¥
  //192.168.7.60/photo-db/¥ <key>
  //192.168.7.60/photo-db/¥UrlSizeLimit = 201 [DWORD]
  //192.168.7.60/photo-db/¥PathSizeLimit = 100 [DWORD]
  //192.168.7.60/photo-db/¥QueryStringSizeLimit = 100 [DWORD]
  //192.168.7.60/photo-db/¥Rule02_QsItem01 = "*=eno" [SZ]
  //192.168.7.60/photo-db/¥Rule02_QsItem03 = "*=0100" [SZ]
  //192.168.7.60/photo-db/¥Rule03_Action = "reJect" [SZ]
  //192.168.7.60/photo-db/¥Rule03_Qs = "*clear*" [SZ]
  //192.168.7.60/photo-db/¥Translations = "|_<00>;_<00><00>"[MULTI_SZ]

guard3 1.26説明書P.14

Guardian@JUMPERZ.NET(1)

- JUMPERZ.NET
- Webサーバ用のIPS
- リバースプロキシとして動作
- Javaで書かれている

http://www.jumperz.net/index.php?i=2&a=0&b=3

IPS (Intrusion Prevention System)

Guardian@JUMPERZ.NET(2)

```
<rule>
name=exploit(requestBody)
type=requestBody
pattern=[¥x00-¥x20¥x7F-¥xFF]
condition=match
case_sensitive=no
log=yes
action=block
command=none
</rule>
<rule>
name=defaced_1
type=responseBody
pattern=owned|r00t|hacked|defaced
condition=match
case_sensitive=no
log=no
action=none
command=none
</rule>
```

不審なアクセスの特徴(1)

- 特殊文字が含まれる
  - <script>alert(document.cookie)</script>
  - x 2>/dev/null;ls -al;#
  - ’or”=’
  - ../../../../etc/passwd%00

不審なアクセスの特徴(2)

- 2xx 3xx **以外のリターンコード**
  - 403 Forbidden
  - 404 Not Found
  - 500 Internal Server Error

パターンマッチ型の問題点

- 防御できない攻撃がある
  - 特殊文字を使わない攻撃
  - リターンコードが200になる攻撃

http://www.example.com/shop.cgi?price=39800

↓

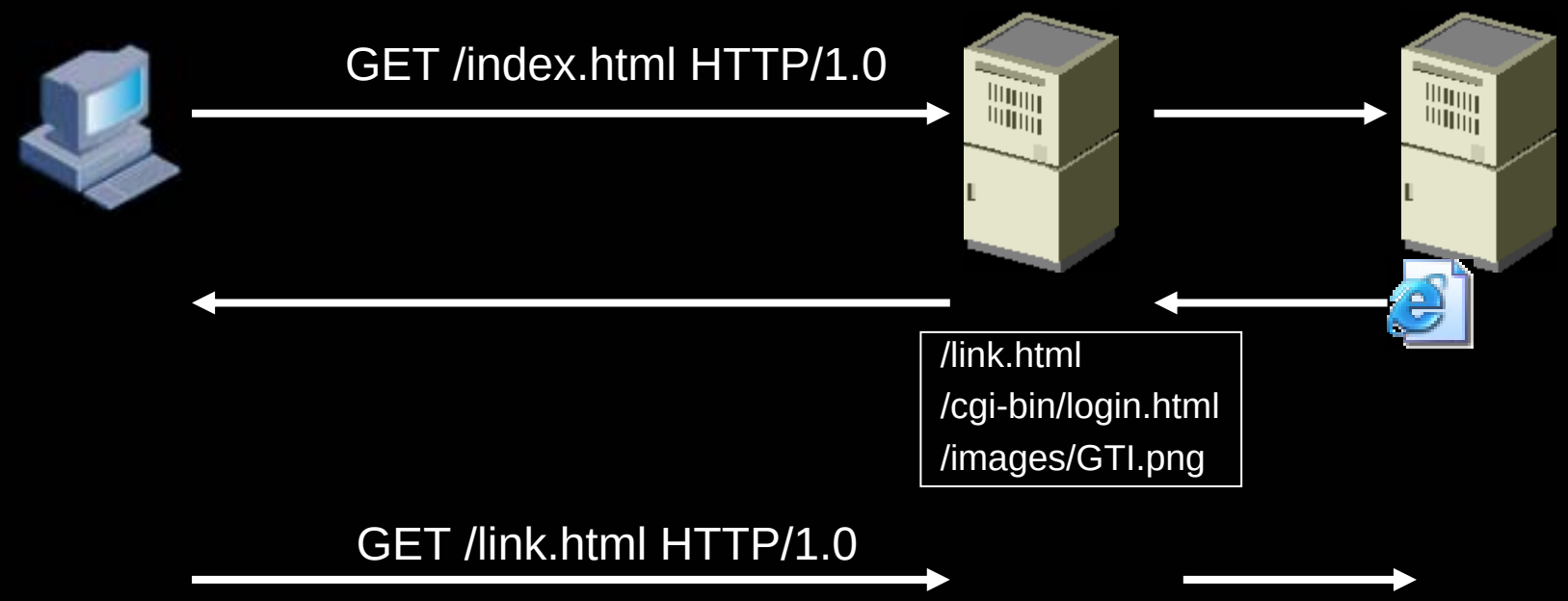
http://www.example.com/shop.cgi?price=12800

http://www.example.com/data/member.csv

AppShield(1)

- 米 Sanctum社
- リバースプロキシとして動作
- 基本は全て拒否
- コンテンツから自動的にルールを作成
- 防御能力はかなり高い（お値段もかなり高い)
- 設定･運用が大変
- http://www.sanctuminc.com/

AppShield(2)


GET /index.html HTTP/1.0
/link.html
/cgi-bin/login.html
/images/GTI.png
GET /link.html HTTP/1.0

防御編まとめ

- 特殊文字を拒否するだけでたいていの攻撃は防げる
- AppShieldはかなり堅い
- すべての攻撃を防御するわけではない

検知編

- 検知といえば、IDS
  - IDSでは検知しきれない

  - 目的が管理者権限とは限らない
  - 攻撃に決まったパターンがない
  - SSLで暗号化されるとお手上げ

- IDS (Intrusion Detection System)

検知方法

- 特殊文字を含んだアクセスを攻撃とみなす
- エラーコードを含んだアクセスを攻撃とみなす
- ログから探し出す

Log Parser(1)

- Ver. 2.1 が IIS 6.0 Resource Kit Toolsに付属している
- Windows Server 2003, Windows XP用
  - Windows2000では、

    iis60rkt.exe /V/a

    として展開し、Log Parserを抜き出せば使える
- IIS のログだけでなく、様々な形式のログで活用できる

Log Parser(2)

- **特殊文字を含んだアクセスの検索**

LogParser.exe “SELECT c-ip AS From,cs-uri-stem AS Path, Count(*) AS Total FROM logs¥ex*.log WHERE cs-uri-stem LIKE ’%¥’%’ GROUP BY c-ip,cs-uri-stem ORDER BY Total DESC" –rtp:-1

```
From          Total Path
------------- ----- ----------------------------------------
192.168.0.234 376   setup=;system( ’ cat%20/etc/passwd ’ )
192.168.0.234 285   file=<script>alert( ’ Vulnerable ’ )</script>
192.168.0.234 96     ’ %0A/bin/cat%20/etc/passwd| ’
192.168.0.234 87    comma=%22;echo%20 ’ ’ ;%20echo%20%60id%20%60;die
```

Log Parser(3)

エラーを頻発させるユーザの検索

LogParser.exe “SELECT c-ip AS From, sc-status AS Status, Count(*) AS Total FROM logs¥ex*.log WHERE NOT sc-status=200 GROUP BY c-ip,sc-status ORDER BY Total DESC”-rtp:-1

```
From            Status Total
--------------- ------ -----
192.168.0.234   404    36297
192.168.0.163   501    24081
192.168.0.234   403    2241
192.168.0.163   403    1448
192.168.0.2     404    798
192.168.0.2     403    340
```

不審なアクセスの特徴(3)

- 同じURLへの頻繁なアクセス
  - /xxx.cgi?id=12345
  - /xxx.cgi?id=12344
  - /xxx.cgi?id=12343
  - /xxx.cgi?id=12342
  - /xxx.cgi?id=12341
  - /xxx.cgi?id=12340
  - /xxx.cgi?id=12339
  - /xxx.cgi?id=12338

Log Parser(4)

- 同じURLへのアクセス

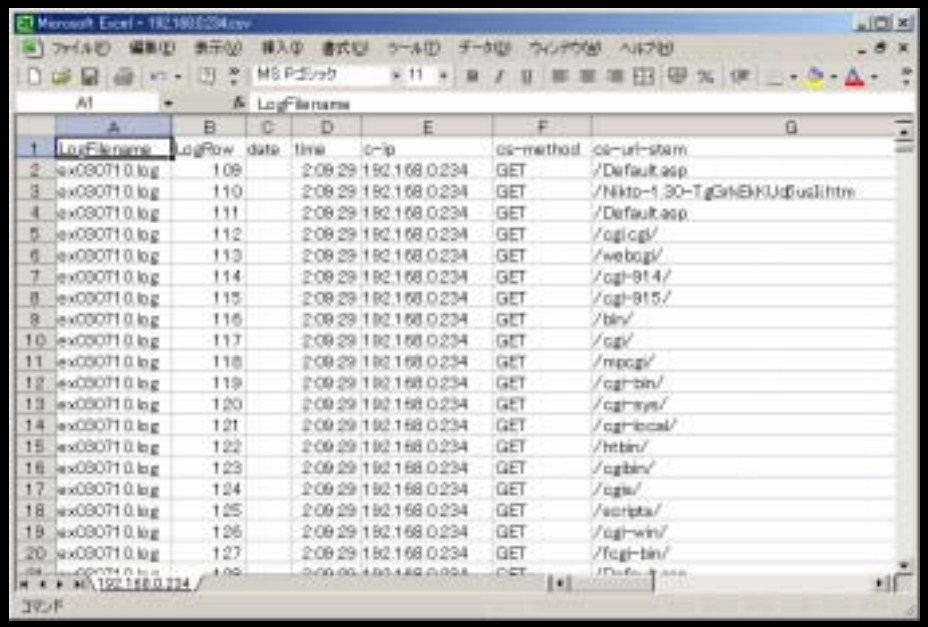
LogParser.exe "SELECT c-ip AS From,cs-uri-stem AS Path, Count(*) AS Total FROM logs¥ex*.log WHERE TO_LOWERCASE(SUBSTR(cs-uri-stem, LAST_INDEX_OF(cs-uri-stem,'.'), STRLEN(cs-uri-stem)))='.asp'GROUP BY c-ip,cs-uri-stem ORDER BY Total DESC" -rtp:-1

```
From           Path          Total
-------------- ------------- ------
192.168.0.222  /hoge2.asp    160017
192.168.0.234  /hoge2.asp    1248
192.168.0.163  /Default.asp  290
192.168.0.158  /hoge.asp     289
```

Log Parser(5)

特定のユーザからのアクセス

LogParser.exe -o:CSV "SELECT * FROM logs¥ex*.log WHERE c-ip='192.168.0.234' ORDER BY LogFilename,LogRow">192.168.0.234.csv



LogFilename	LogRow	date	time	c-ip	cs-method	cs-uri-stem
ex080710.log	109		2:09:29	192.168.0.234	GET	/Default.asp
ex080710.log	110		2:09:29	192.168.0.234	GET	/Nikto-1.30-TgGrNEkKUdfusli.htm
ex080710.log	111		2:09:29	192.168.0.234	GET	/Default.asp
ex080710.log	112		2:09:29	192.168.0.234	GET	/cgi-cgi/
ex080710.log	113		2:09:29	192.168.0.234	GET	/webcgi/
ex080710.log	114		2:09:29	192.168.0.234	GET	/cgi-914/
ex080710.log	115		2:09:29	192.168.0.234	GET	/cgi-915/
ex080710.log	116		2:09:29	192.168.0.234	GET	/bin/
ex080710.log	117		2:09:29	192.168.0.234	GET	/cgi/
ex080710.log	118		2:09:29	192.168.0.234	GET	/mpcgi/
ex080710.log	119		2:09:29	192.168.0.234	GET	/cgi-bin/
ex080710.log	120		2:09:29	192.168.0.234	GET	/cgi-sys/
ex080710.log	121		2:09:29	192.168.0.234	GET	/cgi-local/
ex080710.log	122		2:09:29	192.168.0.234	GET	/htbin/
ex080710.log	123		2:09:29	192.168.0.234	GET	/cgbin/
ex080710.log	124		2:09:29	192.168.0.234	GET	/cgis/
ex080710.log	125		2:09:29	192.168.0.234	GET	/scripts/
ex080710.log	126		2:09:29	192.168.0.234	GET	/cgi-win/
ex080710.log	127		2:09:29	192.168.0.234	GET	/fcgi-bin/

まだ見つけられない攻撃

- POSTメソッドを使っている場合はどうするか

  ⇩

  HTTPリクエスト丸ごと保存しておく？

- アクセス数が少ないと見逃す
- セッションハイジャック

検知編まとめ

- 簡単に見つかるものは簡単に守れる
- 膨大な手間・時間・リソースを使ってみつける
- 絶対に見つけられない攻撃もある

まとめ

- 脆弱なアプリケーションは守りきれない
- 攻撃を事後に見つけるのも困難
- 設計・プログラミング・テストの段階でセキュリティを意識した構築を

参考資料

- IPA ISEC セキュアプログラミング講座
  http://www.ipa.go.jp/security/awareness/vendor/programming/
- 安全なWebアプリ開発40箇条の鉄則
  http://java-house.jp/~takagi/paper/idg-jwd2003-takagi-dist.pdf
- 高木浩光@茨城県つくば市 の日記
  http://d.hatena.ne.jp/HiromitsuTakagi/20030928#p1
- Forensic Log Parsing with Microsoft's LogParser
  http://www.securityfocus.com/infocus/1712
- Without a Trace: Forensic Secrets for Windows Servers
  http://www.blackhat.com/html/win-usa-04/bh-win-04-speakers.html