SW-Entwurf von selbstverteidigenden Systemen

Inhalt

- Einführung
- Ablauf Angriff aus dem Internet
- Reaktion auf Angriffe
- Erkennen von Angriffen
- Fake-Daten / Honey-Data
- Error-Trigger
- Zusammenfassung

Einführung

Dieser Vortrag zeigt, wie man Angreifer aus dem Internet am Diebstahl von Daten hindern kann.

Dieser Vortrag geht nicht auf andere Angreifertypen wie beispielsweise interne Mitarbeiter ein, da diese andere Methoden verwenden. Die hier gezeigten Methoden erkaufen sich den Schutz durch eine Downtime der Systeme. Der Kunde muss entscheiden, ob das ein möglicher Weg darstellt.

Dieser Ansatz sollte zusammen mit anderen Architekturansätzen (Verschlüsselung, Auditing, …) implementiert werden.

SQL Injection

SQL Injection ist einer der häufigsten und gefährlichsten Fehler in Webanwendungen.

Ursache sind die oft die schnelle Entwicklung von Anwendungen ohne Security-Testen, fehlendes Wissen zur Vermeidung von SQL Injection, Fehler in Frameworks, …

Die SQL Injection Fehler werden in der Regel verwendet, Daten zu stehlen.

Wie läuft ein Angriff gegen Datenbanken aus dem Internet ab?

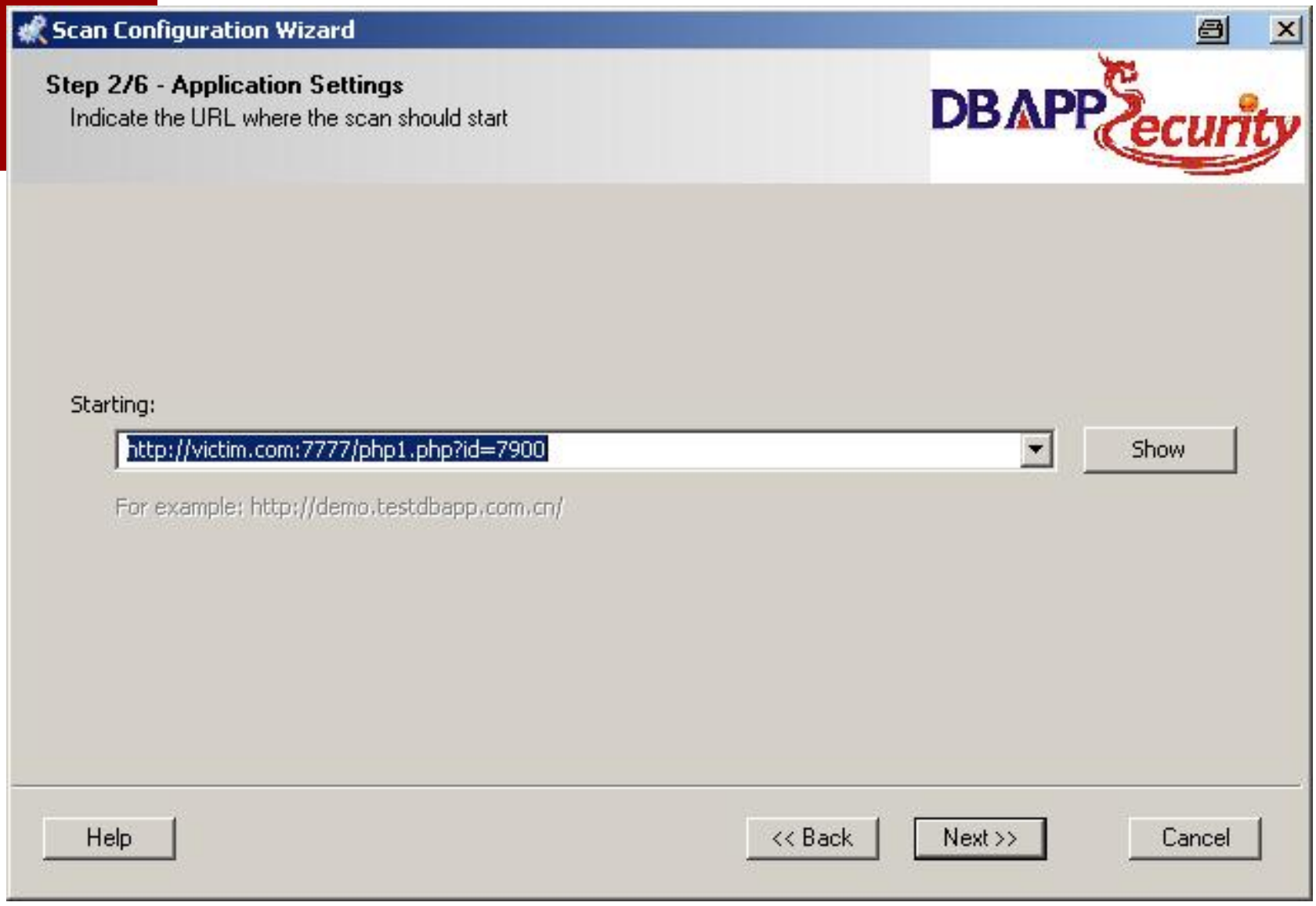
1. Ziel  finden

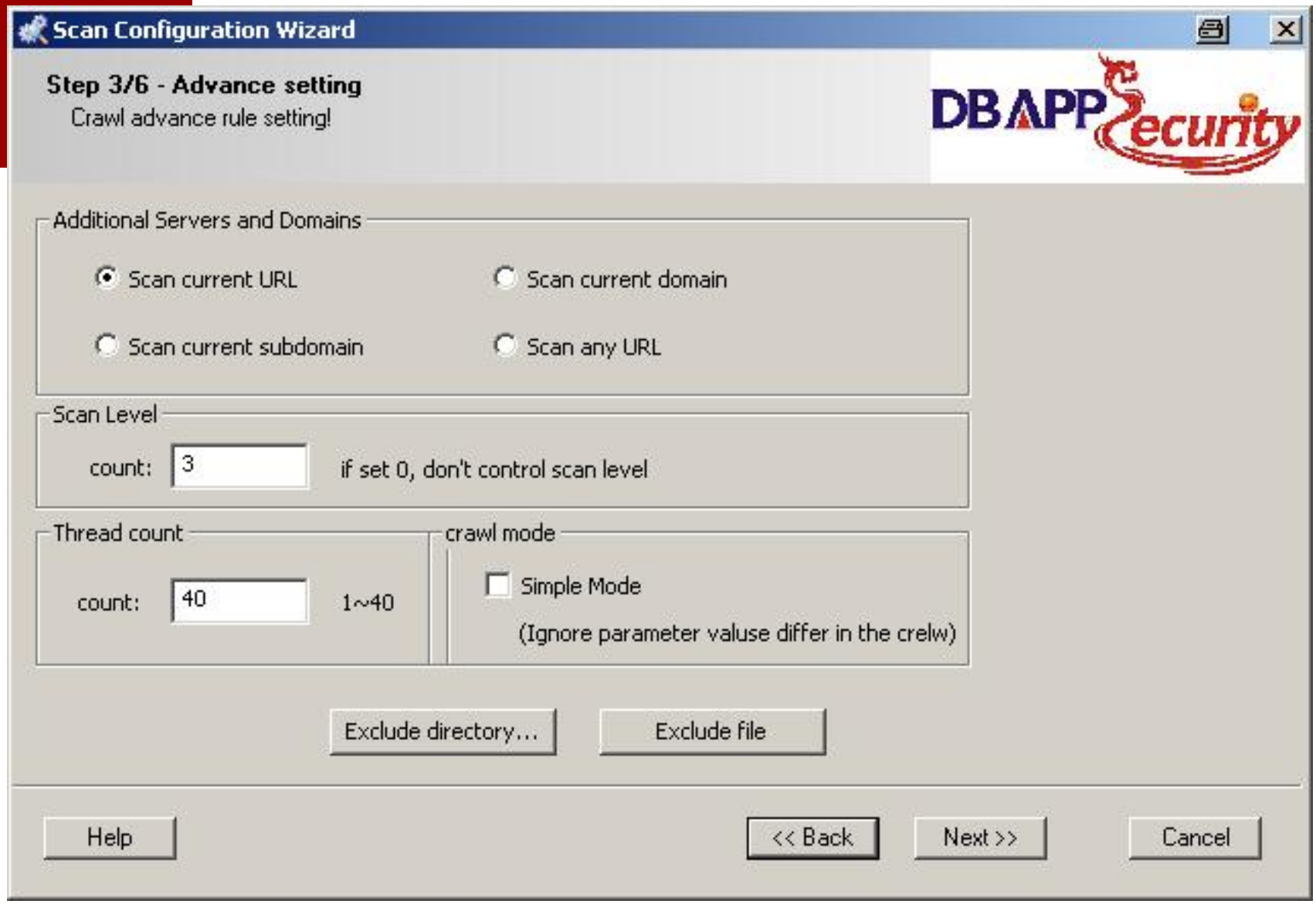
Angreifer / Hacker verwendet Google Hacking oder das Scanning der Internetpräsens/Webseiten eines Unternehmens/Organisation, um Angriffspunkte zu finden.

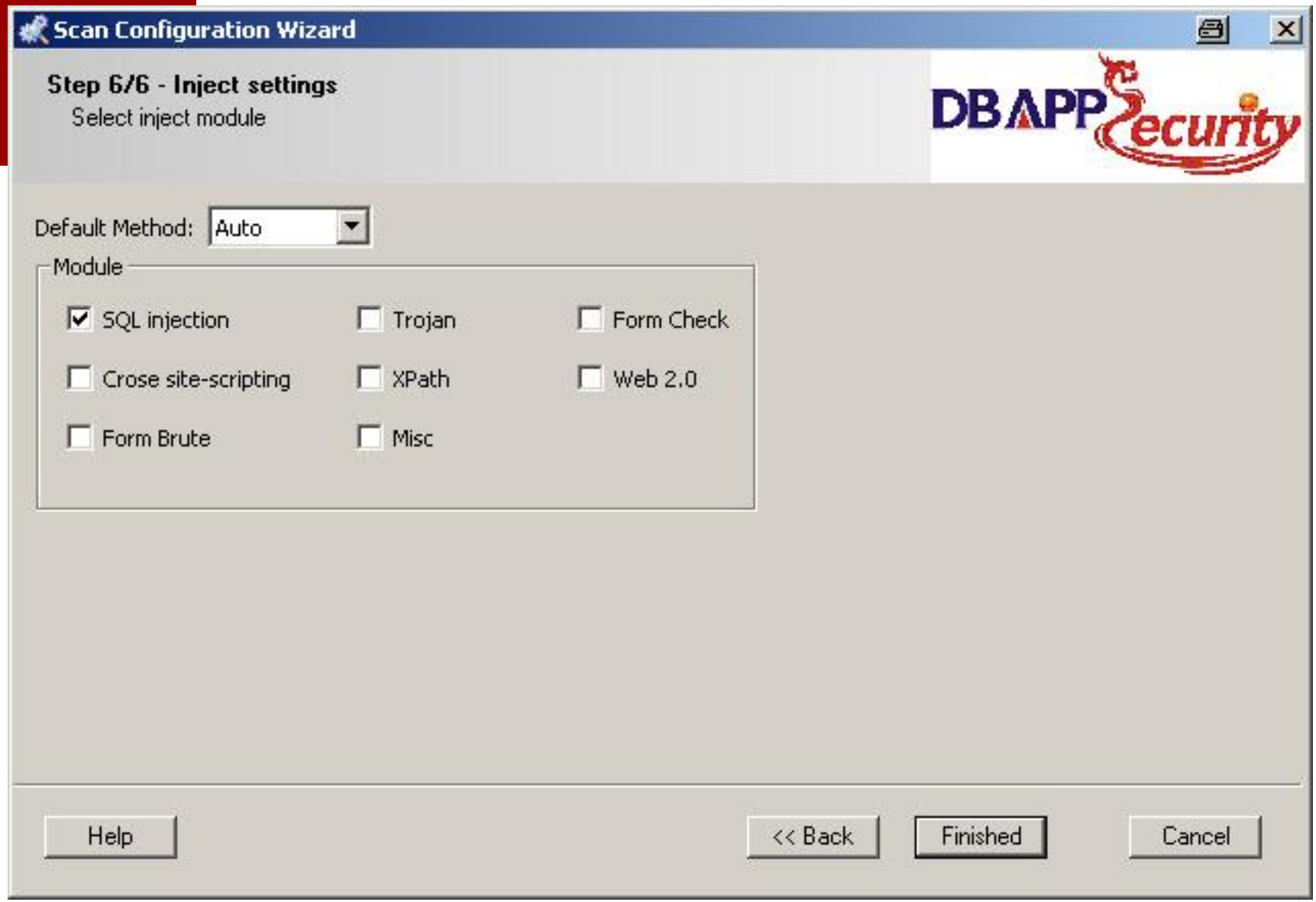
Typischerweise werden als Werkzeuge raubkopierte Versionen von Security-Software (Netsparker, Havij, AppScan, …) verwendet.

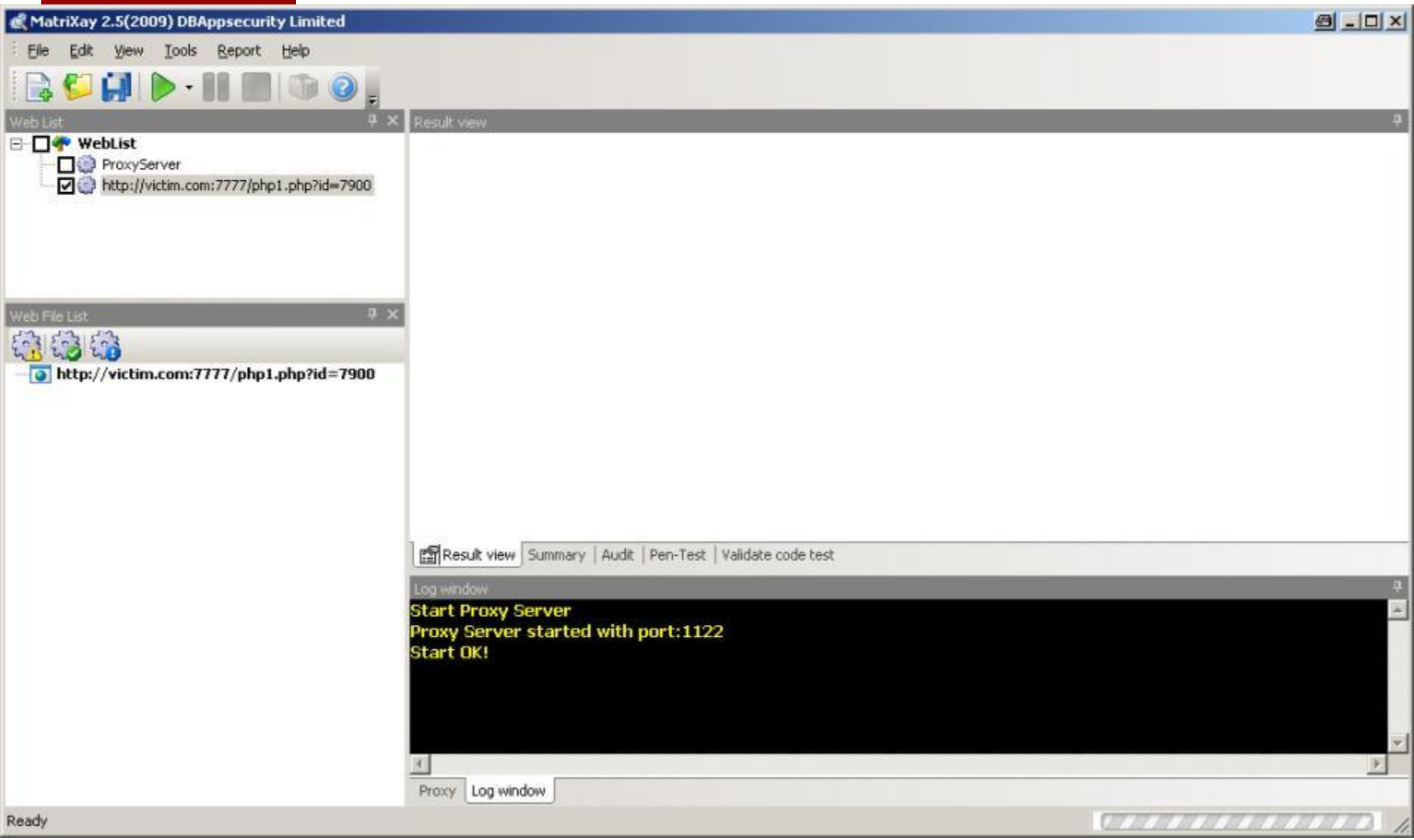
Einige der von Hackern verwendeten Werkzeuge

- Havij
- Netsparker
- HP Webinspect™
- HP Scrawlr
- IBM Rational Appscan
- Matrixay
- Pangolin
- SQLMap
- BSQLHacker
- …

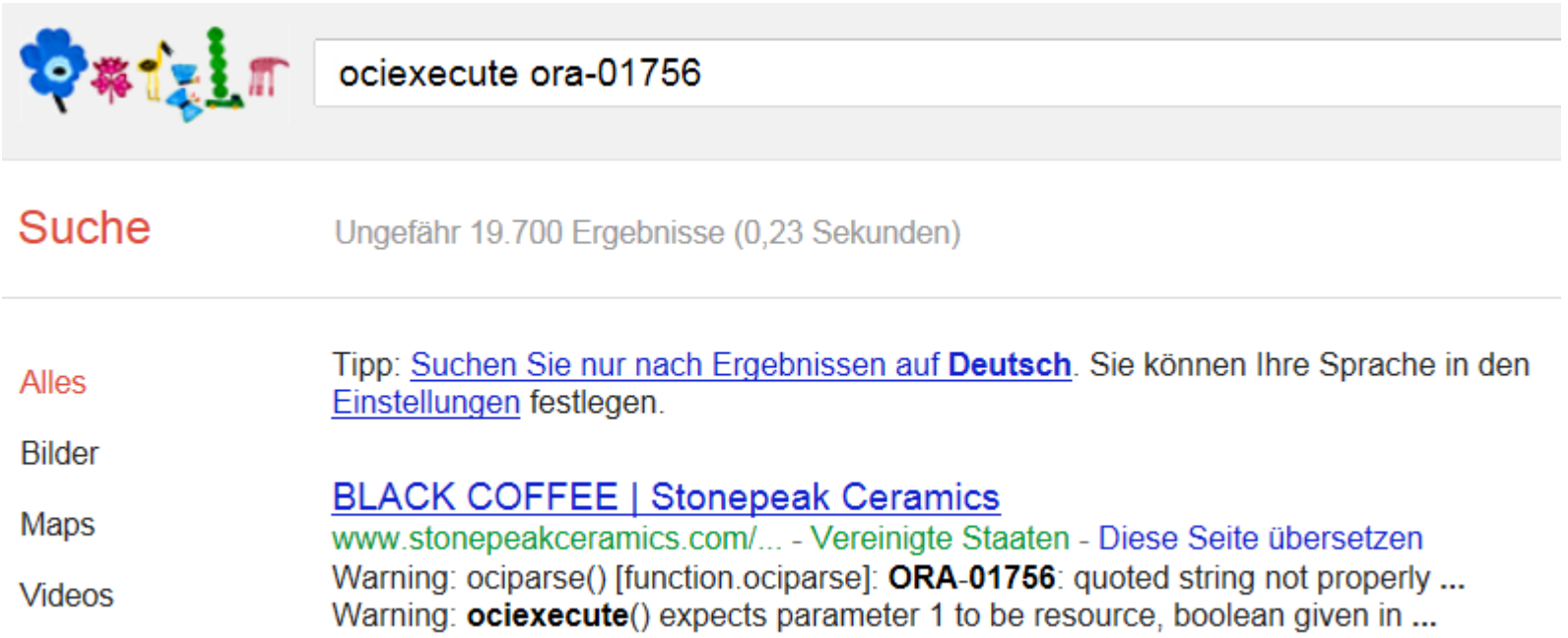

Scan Configuration Wizard
Step 2/6 - Application Settings
Indicate the URL where the scan should start
DBAPPSecurity
Starting:
http://victim.com:7777/php1.php?id=7900
Show
For example: http://demo.testdbapp.com.cn/
Help
<< Back
Next >>
Cancel


Scan Configuration Wizard
Step 3/6 - Advance setting
Crawl advance rule setting!
DBAPPSecurity
Additional Servers and Domains
Scan current URL
Scan current domain
Scan current subdomain
Scan any URL
Scan Level
count:
3
if set 0, don't control scan level
Thread count
count:
40
1~40
crawl mode
Simple Mode
(Ignore parameter valuse differ in the crelw)
Exclude directory...
Exclude file
Help
<< Back
Next >>
Cancel


Scan Configuration Wizard
Step 6/6 - Inject settings
Select inject module
DBAPPSecurity
Default Method:
Auto
Module
SQL injection
Trojan
Form Check
Crose site-scripting
XPath
Web 2.0
Form Brute
Misc
Help
<< Back
Finished
Cancel

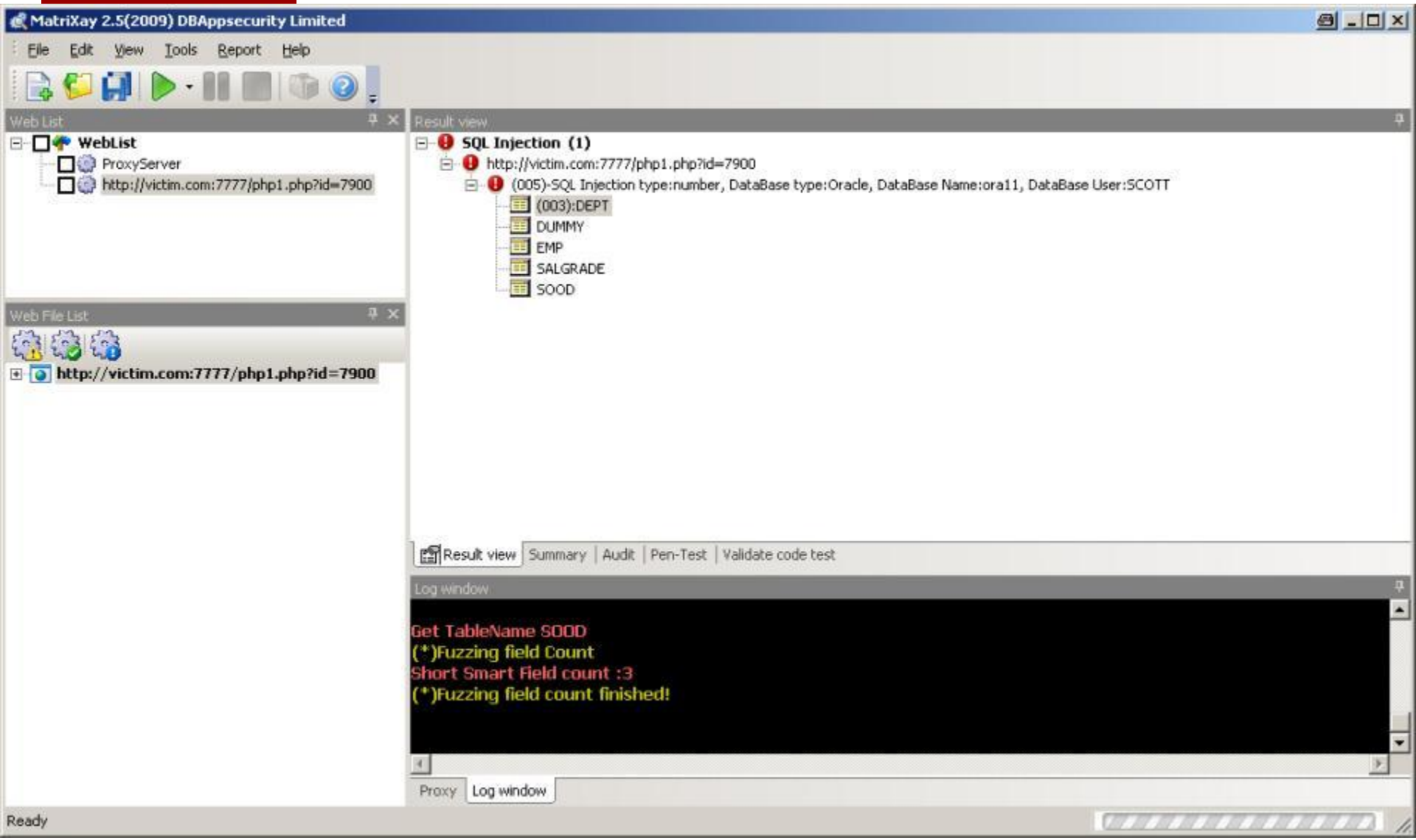

MatriXay 2.5(2009) DBAppsecurity Limited
File
Edit
View
Tools
Report
Help
Web List
WebList
ProxyServer
http://victim.com:7777/php1.php?id=7900
Web File List
http://victim.com:7777/php1.php?id=7900
Result view
Result view
Summary
Audit
Pen-Test
Validate code test
Log window
Start Proxy Server
Proxy Server started with port:1122
Start OK!
Proxy
Log window
Ready

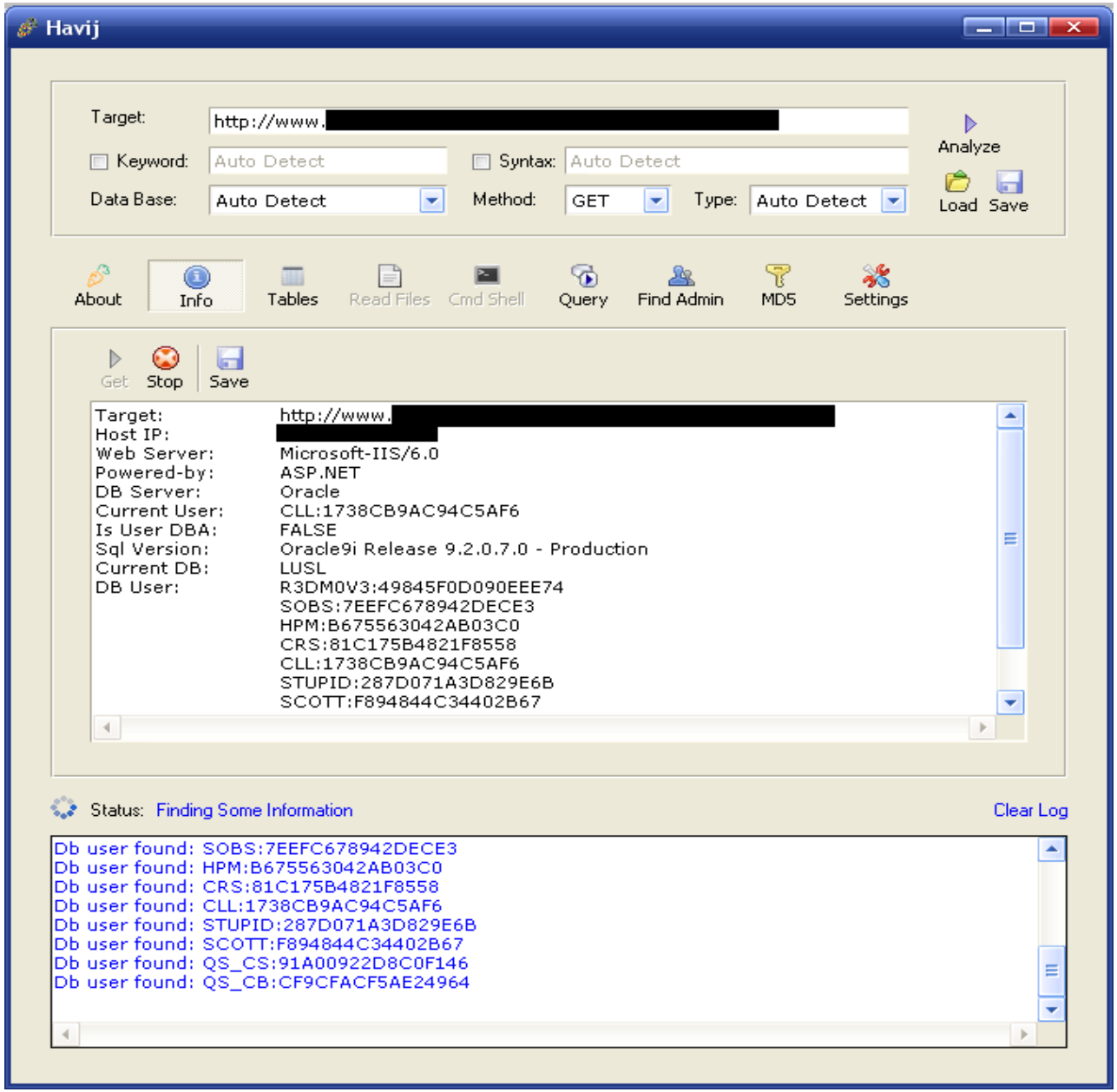
Google Hacking



2. Daten auswählen

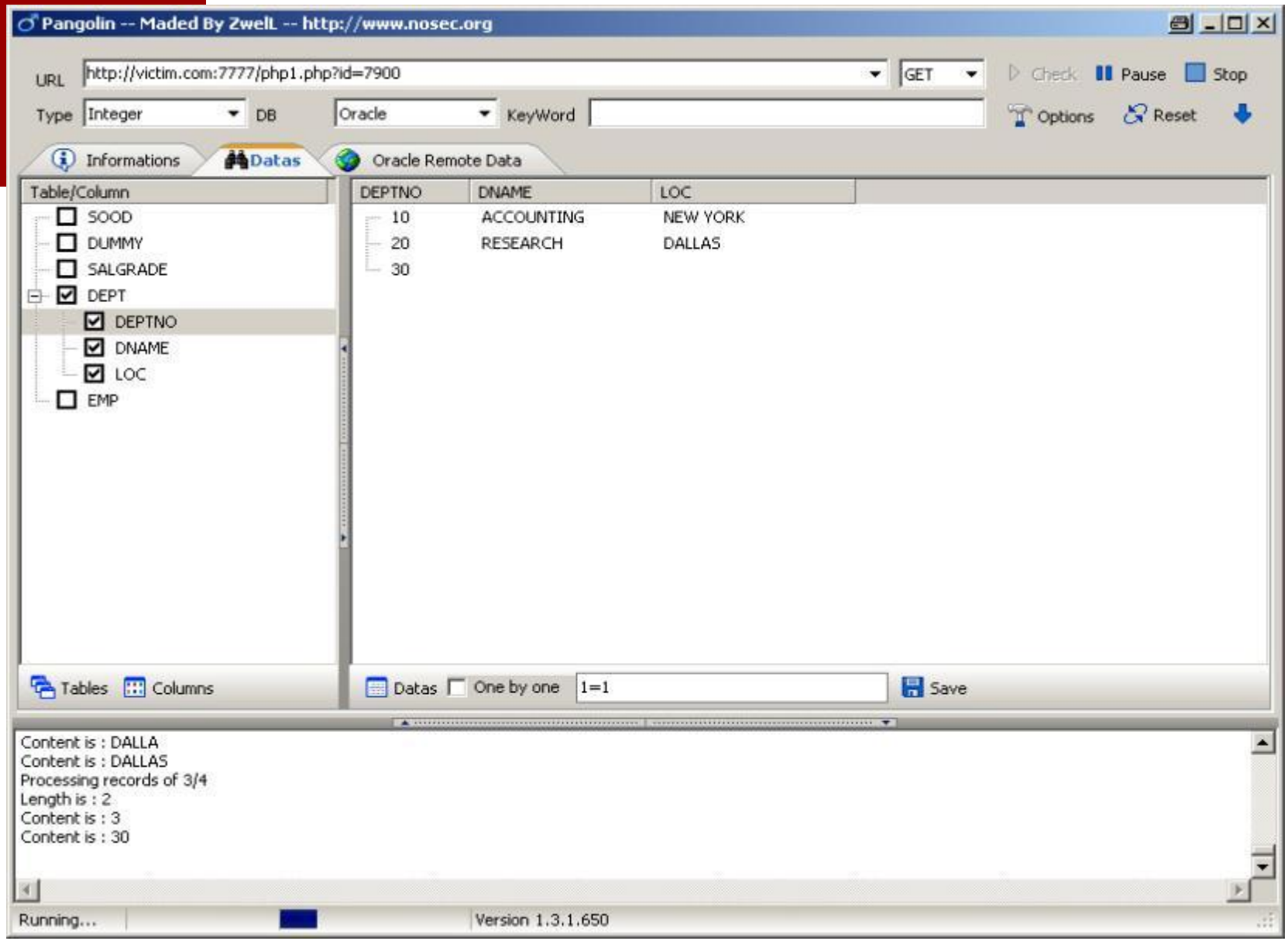
Der Angreifer sucht die interessanten Daten (toolgesteuert) anhand von Tabellen und/oder Spaltennamen aus


MatriXay 2.5(2009) DBAppsecurity Limited
File Edit View Tools Report Help
Web List
WebList
ProxyServer
http://victim.com:7777/php1.php?id=7900
Web File List
http://victim.com:7777/php1.php?id=7900
Result view
SQL Injection (1)
http://victim.com:7777/php1.php?id=7900
(005)-SQL Injection type:number, DataBase type:Oracle, DataBase Name:ora11, DataBase User:SCOTT
(003):DEPT
DUMMY
EMP
SALGRADE
SOOD
Result view Summary Audit Pen-Test Validate code test
Log window
Get TableName SOOD
(*)Fuzzing field Count
Short Smart Field count :3
(*)Fuzzing field count finished!
Proxy Log window
Ready


Havij
Target:
http://www.[redacted]
Analyze
Keyword:
Auto Detect
Syntax:
Auto Detect
Data Base:
Auto Detect
Method:
GET
Type:
Auto Detect
Load
Save
About
Info
Tables
Read Files
Cmd Shell
Query
Find Admin
MD5
Settings
Get
Stop
Save
Target: http://www.[redacted]
Host IP: [redacted]
Web Server: Microsoft-IIS/6.0
Powered-by: ASP.NET
DB Server: Oracle
Current User: CLL:1738CB9AC94C5AF6
Is User DBA: FALSE
Sql Version: Oracle9i Release 9.2.0.7.0 - Production
Current DB: LUSL
DB User: R3DM0V3:49845F0D090EEE74
SOBS:7EEFC678942DECE3
HPM:B675563042AB03C0
CRS:81C175B4821F8558
CLL:1738CB9AC94C5AF6
STUPID:287D071A3D829E6B
SCOTT:F894844C34402B67
Status: Finding Some Information
Clear Log
Db user found: SOBS:7EEFC678942DECE3
Db user found: HPM:B675563042AB03C0
Db user found: CRS:81C175B4821F8558
Db user found: CLL:1738CB9AC94C5AF6
Db user found: STUPID:287D071A3D829E6B
Db user found: SCOTT:F894844C34402B67
Db user found: QS_CS:91A00922D8C0F146
Db user found: QS_CB:CF9CFACF5AE24964

3. Daten herunterladen

Die Daten werden automatisiert heruntergeladen und als CSV Datei abgespeichert.


Pangolin -- Maded By ZwelL -- http://www.nosec.org
URL
http://victim.com:7777/php1.php?id=7900
GET
Check
Pause
Stop
Type
Integer
DB
Oracle
KeyWord
Options
Reset
Informations
Datas
Oracle Remote Data
Table/Column
SOOD
DUMMY
SALGRADE
DEPT
DEPTNO
DNAME
LOC
EMP
DEPTNO
DNAME
LOC
10
ACCOUNTING
NEW YORK
20
RESEARCH
DALLAS
30
Tables
Columns
Datas
One by one
1=1
Save
Content is : DALLA
Content is : DALLAS
Processing records of 3/4
Length is : 2
Content is : 3
Content is : 30
Running...
Version 1.3.1.650

Und schon sind die Daten weg…

Vom Beginn des Scans bis zum Herunterladen vergehen oft weniger als 5 Minuten…

Frage: Was würden Sie tun, wenn Sie einen laufenden Angriff entdecken?

- Nichts
- Manager informieren
- Applicationsowner informieren
- Datenbank herunterfahren
- Account sperren
- …

Erfahrung

- In der Regel sind die Firmen/Organisationen (vor allem Nachts bzw. am Wochenende) zu langsam, um auf solche Angriffe zu reagieren…
- Zuständigkeiten sind nicht geregelt
- Anwendungen nicht darauf vorbereitet

Lösung:

Selbstverteidigende Systeme

Design-Vorgabe

- System muss in der Lage sein, Angriffe von außen zu erkennen
- System muss darauf entsprechend reagieren können
- Mechanismus muss schnell (im Fehlerfall) deaktiviert werden können

Erkennen von Angriffen

- Fast jeder SQL Injection Angriff verursacht in echten Leben Fehler in der Datenbanken
- Ausnahme sind kommerzielle Anwendungen (z.B. CMS) mit bekannten/vordefinierten Exploits, da nicht rumprobiert werden muss
- Auch das Suchen und der Zugriff auf kritische Fake-Daten kann als Angriff erkannt werden

Fake-Daten (Honey-Data)

- Das Anlegen und Überwachen (z.B. mit VPD) von der Anwendung nicht verwendeten Daten kann helfen, Angriffe zu erkennen
- Angreifer greifen oft auf die Views ALL_TAB_COLUMNS bzw. ALL_TABLES um die interessanten Daten (z.B. PASSWORD, CREDITCARD, …) zu finden
- Nach dem Finden der Tabellen, werden diese in der Regel heruntergeladen
- Eine VPD-Regel auf der Tabelle mit Fake-Daten kann entsprechend Reagieren und der Benutzer, der den Zugriff versucht, sperren

Fake-Daten (Honey-Data)

```
-- Honeytabelle erzeugen

create table app.kundendaten (username varchar2(30), password varchar2(30));

-- Honeytable mit Daten füllen
insert into app.kundendaten values ('WEBUSER','WEBUSER01');
insert into app.kundendaten values ('WEBADM','ADMADM01');
insert into app.kundendaten values ('WEBREAD','READUSER01');

-- predicate function with anlegen
create or replace function perfcheck (pv_schema in varchar2, pv_object in
varchar2)
return varchar2 as
 begin

dbms_output.put_line('Email an Security versenden...');
-- code zum Senden einer Email einfuegen
-- return always true. Attacker will see all results

   return '1=1';
end;
/
```

Erkennen von SQL Injection Angriffen über Fehler

- Je nach verwendeter Angriffsmethode (UNION, Abfrage erweitern, Fehlermeldung erzeugen, …) wird ein spezifischer Oracle-Fehler
- Z.B. ORA-01789: query block has incorrect number of result columns

Typischer SQL Injection Angriff

Ursprünglicher Befehl

select custname, custid, custorder from customer;

Befehl mit “erweiterter” Funktionalität

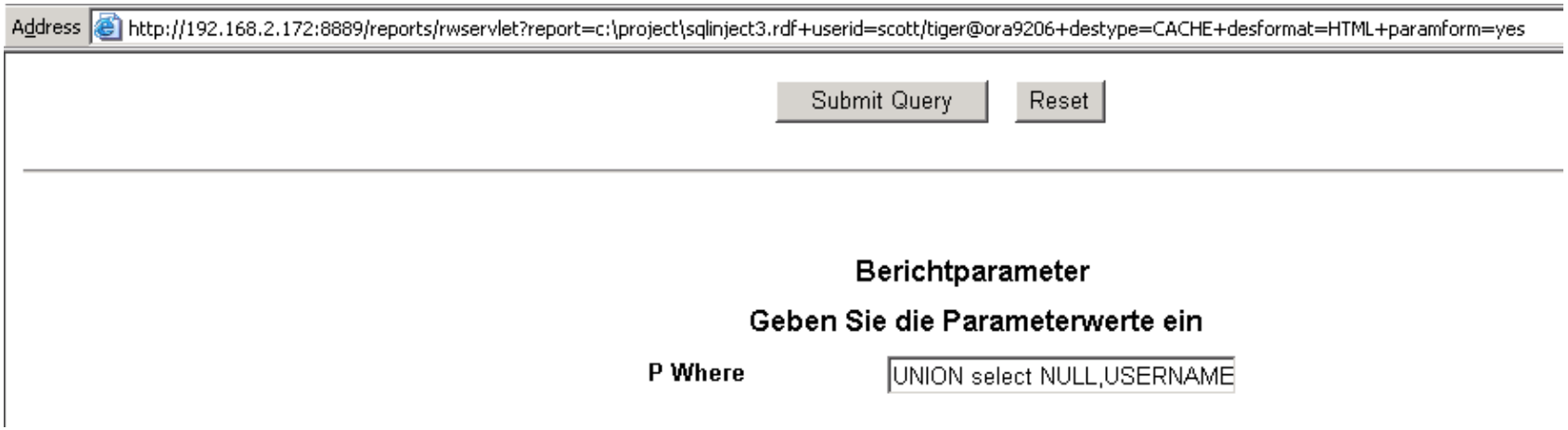
select custname, custid, custorder from customer
union
select username, null, password from dba_users;

Typischer SQL Injection Angriff

http://myserver:8889/reports/rwservlet?report=sqlinject3.rdf+userid=scott/tiger@ora9206+destype=CACHE+desformat=HTML

Typischer SQL Injection Angriff

Typischer SQL Injection Angriff



Typischer SQL Injection Angriff

Typische Fehlermeldung:

ERROR at line 1:
ORA-01789: query block has incorrect number of result columns

➔ Angreifer (oder Tool) verwendet NULL, um ein korrektes SQL Statement zu erzeugen

Typischer SQL Injection Angriff

Address http://192.168.2.172:8889/reports/rwservlet?

SQL Injection

Empno	Ename	Sal
7369	SMITH	800
7499	ALLEN	1600
7521	WARD	1250
7566	JONES	2975
7654	MARTIN	1250
7698	BLAKE	2850
7782	CLARK	2450
7788	SCOTT	3000
7839	KING	5000
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	950
7902	FORD	3000
7934	MILLER	1300
	ANONYMOUS	
	CTXSYS	
	DBSNMP	
	HR	
	MDSYS	
	ODM	

Erkennen von SQL Injection Fehlern

Fehlernr	Fehlermeldung	Auslöser
ORA-00900	invalid SQL statement	
ORA-00906	missing left parenthesis	
ORA-00907	missing right parenthesis	
ORA-00911	invalid character	z.B. PHP MAGIC_QUOTES_GPC sind aktiviert und es wird versucht ein single quote zu injected
ORA-00917	missing comma	
ORA-00920	invalid relational operator	
ORA-00923	FROM keyword not found where expected	
ORA-00933	SQL command not properly terminated	
ORA-00970	missing WITH keyword	
ORA-01031	insufficient privileges	Versuch der Privilegieneskalation
ORA-01476	divisor is equal to zero	Versuch von Blind SQL Injection (z.b. vom Tool sqlmap)
ORA-01719	outer join operator not allowed in operand of OR or IN	
ORA-01722	invalid number	Enumerierung mit rownum und aktuelle rownum existiert nicht
ORA-01742	comment not properly terminated	inline comment, z.B. optimizer hint, wird nicht richtig mit */ terminiert

Erkennen von SQL Injection Fehlern

Fehlernr	Fehlermeldung	Auslöser
ORA-01742	comment not properly terminated	inline comment, z.B. optimizer hint, wird nicht richtig mit */ terminiert
ORA-01756	quoted not properly terminated	single quote wird nicht richtig terminiert
ORA-01789	query block has incorrect number of result columns	Versuch UNION SELECT zu verwenden
ORA-01790	expression must have same datatype as corresponding	Versuch UNION SELECT zu verwenden
ORA-24247	network access denied by access control list	Oracle ACL verhindert Verwendung von UTL_INADDR oder ähnliche Funktionen
ORA-29257	Host %S unknown	Versuch SQL Injection mittels utl_inaddr
ORA-29540	Class does not exist	Versuch utl_inaddr zu Verwenden, aber Java ist nicht installiert (Oracle 10 oder älter)
ORA-31011	XML parsing failed	Versuch SQL Injection mittels xmltype
ORA-19202	Error occurred in XML processing	Versuch SQL Injection mittels extractvalue

Reaktion auf Fehler

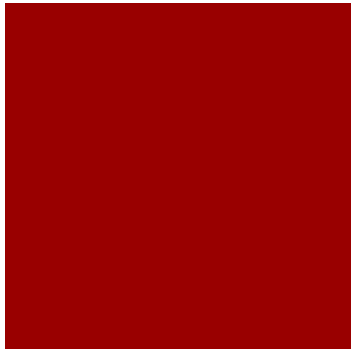
- Ein Oracle Error-Trigger kann auf diese Fehler reagieren
  - (Nur) Mitprotokollieren
  - Datenbank-Kennung sperren
  - Diese/Alle Sessions beenden
  - Email an Manager-on-Duty/DBA/Security Abteilung senden
- Zur Minimierung der Nicht-Verfügbarkeit, sollten verschiedene Kennungen (z.B. Internet, Intranet, Android, ..) verwendet werden.
- Das Sperren der Internet-Anwendung sollte die Verwendung aus dem Intranet nicht behindern
- Nur Fehler vom Applikation Server sollte eine Reaktion auslösen (d.h. ORA-01756 von sqlplus wird ignoriert)

```
CREATE OR REPLACE TRIGGER after_error
 AFTER SERVERERROR ON DATABASE
 DECLARE
 sql_text ORA_NAME_LIST_T;
 v_stmt CLOB;        -- SQL statement causing the problem
 n NUMBER;           -- number of junks for constructing the sql statement causing the
error
 v_program VARCHAR2(64);
 v_serial number;
 v_sid number;
BEGIN
-- Version 1.00
select program,serial#,sid into v_program,v_serial,v_sid from v$session where
sid=sys_context('USERENV', 'SID');
 -- construct the sql text
 n := ora_sql_txt(sql_text);
 --
 IF n >= 1
 THEN
 FOR i IN 1..n LOOP
 v_stmt := v_stmt || sql_text(i);
 END LOOP;
 END IF;
 --
 FOR n IN 1..ora_server_error_depth LOOP
```

```
 IF (lower(v_program) = 'iis.exe')
   and (ora_server_error(n) in
('942','900','906','907','911','917','920','923','933','970','1031','1476','1719','1722','1742','1756','17
89','1790','19202','24247','29257','29540','31011'))
 THEN
   -- Es wurde ein moeglicher Angriff erkannt
   -- 1.  Eintrag mitprotokollieren

   -- 2. Zustaendigen per Email informieren
   -- send_email (z.b. via utl_smtp muss noch implementiert werden

    -- 3. Kennung sperren, die die SQL Injection ausgeloest hat
     execute immediate ('ALTER USER /* Error_Trigger */
'" | |sys_context('USERENV','SESSION_USER') | |'" account lock');

        -- 4. Session beenden, die die SQL Injection ausgeloest hat
     execute immediate ('ALTER SYSTEM /* Error_Trigger */ KILL SESSION
'" | |v_sid | |',' | |v_serial | |'" account lock');
alter system kill session 'session-id,session-serial'

 END IF;

 END LOOP;
 --
END after_error;
/
```

Reaktion

- Nach dem Sperren des Accounts kann der Webdienst nicht mehr (vom Internet) verwendet werden
- Email sollte an das Operating / Manager of Duty gesendet werden
  - Fehler analysieren ('or 1=1 oder O‘Leary)
  - Bei Fehlalarm den Account entsperren
  - Bei echtem Alarm Angreifer identifizieren (Amateur, Profi, Tool-User, …)
  - Eventuell IP-Ranges sperren (nicht-Deutschland sperren)
  - Fehler in Software korrigieren bzw. Workaround suchen.



Fragen & Antworten?

Danke

- Kontakt:

Red-Database-Security GmbH

Bliesstr. 16

D-.66538 Neunkirchen

Germany