ATK gegen WEKA

Technische Beweisführung

Version 2007-07-30

Marc Ruef

1. Einführung

1.1. Ausgangslage

Marc Ruef wirft dem deutschen Unternehmen WEKA Business Information GmbH & Co. KG. (http://www.weka.de) vor, unrechtmässig Programmteile aus der Software Attack Tool Kit (ATK) übernommen zu haben [Ruef 2006a]. Der Geschäftsbereich Interest vertreibt seit mindestens Oktober 2005 in kommerzieller Weise eine Software mit dem Namen Interest Securitix Scanner.

Der Streitpunkt sowie das vorliegende Dokument berufen sich auf jene Software, wie sie nachfolgend in tabellarischer Form aufgelistet ist. Sämtliche Analysen wurden an den gegebenen Versionen durchgeführt. Der Interest Securitix Scanner wird in dieser Dokumentation schlicht mit Securitix und das Attack Tool Kit mit ATK referenziert. Die beiden Streitparteien werden nachfolgend WEKA und Ruef genannt.

	Produkt	Erscheinungsdatum	Entwickler
S1	Interest Securitix Scanner 1.6	Oktober 2005	WEKA/Feil
S2	Interest Securitix Scanner 1.8	Juni 2006	WEKA/Feil
A1	Attack Tool Kit (ATK) 4.1	Februar 2006	ATK/Ruef
A2	Attack Tool Kit (ATK) 1.0	Februar 2004	ATK/Ruef

Tabelle 1: Die analysierten Produkte

Die jeweiligen Produkte wurden von ihren offiziellen Quellen bezogen. Interest hat den Securitix Scanner (S1 und S2) bis zum Juli 2006 als freie Trial-Version auf der offiziellen Webseite unter http://www.securityscanner.de angeboten. Das Attack Tool Kit (A1 und A2) wird seit Februar 2004 zum freien Download auf der Projektseite http://www.computec.ch/projekte/atk/ bereitgestellt. Neben einer kompilierten Version ist ebenfalls der Quelltext dieser unter der General Public License (GPL) bereitgestellt worden.

1.2. Ziel dieses Dokuments

Aufgrund der wiederholten Zurückweisung der an WEKA herangetragenen Vorwürfe und den gegebenen Falschaussagen soll in diesem Dokument eine technische Analyse und damit eine Beweisführung auf dieser Ebene gewährleistet werden. Mit dem Vergleich der beiden Software-Produkte ATK von Ruef und Securitix von WEKA sollen die offensichtlichen Übernahmen als solche ausgewiesen werden.

1.3. Vorgehen der Analyse

Ruef liegen zwei unterschiedliche Versionen von Securitix in kompilierter Form vor. Ebenso sind ihm als Entwickler sämtliche Versionen in quelloffener und kompilierter Form des ATK zugänglich. Die Analyse findet dabei auf zwei Ebenen statt:

- **Graphical User Interface**: In erster Linie soll eine Gegenüberstellung der beiden Applikationen von blossem Auge erfolgen. Durch das zeitgleiche Aufstarten beider Applikationen und der Dokumentation mittels Screenshots werden die unrechtmässig übernommenen Teile ausgemacht. (Kapitel 3.1)

- **Binary Analysis**: Desweiteren wird durch ein Reverse Engineering des Securitix dessen interne Funktionsweise offengelegt. Durch eine bitweise Überprüfung der beiden Anwendungen sollen die übernommenen Teile zweifelsfrei identifiziert werden. (Kapitel 3.2)

Die gegebenen Daten, die als unrechtmässig übernommen ausgemacht werden konnten, werden für eine formale Analyse aufbereitet. Hierbei werden Berechnungen angestrebt und Argumente vorgetragen, die beweisen sollen, dass es sich bei der Vielzahl der übernommenen Teile nicht um ein Versehen oder einen Zufall handeln kann. (Kapitel 4)

1.4. Unabhängigkeit der Analyse

Dieses Dokument sowie die zugrundeliegende Analyse wurden durch Marc Ruef durchgeführt. Dieser ist der Entwickler des Attack Tool Kit, damit eine der beiden Streitpartien und deshalb **nicht** als unabhängiger Gutachter anzusehen. Trotzdem wird dieses Schreiben bei einer allfälligen Klage gegen WEKA dem Gericht vorgelegt werden wollen.

Durch das detaillierte Vortragen und Untermauern der Vorwürfe wird es aber auch Dritten ermöglicht, eine Verifikation der Gegebenheiten, sofern Zugriff auf die betroffenen Produkte gewährt wird, durchzuführen. Eine solche Nachprüprüfung ist sogar ausdrücklich durch Ruef erwünscht, hilft sie doch eventuell nur dabei, den Sachverhalt noch offensichtlicher darlegen zu können.

Desweiteren ist ein externes Gutachten eines unabhängigen Experten in Arbeit. Die Resultate dessen sollten sich mit den Ausführungen dieses Dokuments decken. Ob und inwiefern dieses Schriftstück ebenfalls frühzeitig der Öffentlichkeit vorgelegt wird, ist zum jetzigen Zeitpunkt noch nicht klar. Details zum Fall werden umfassend auf http://www.computec.ch veröffentlicht.

2. Zusammenfassung

2.1. Zusammenfassung

Die vorliegende Analyse zeigt sehr deutlich, dass auch mit limitierten technischen Mitteln sowie ausschliesslich grundlegendem Verständnis für Computersysteme die unerlaubte Kopie gewisser Teile des Attack Tool Kit Project (ATK) durch den Interest Securitix Scanner identifizierbar sind. Selbst durch das Betrachten der beiden Anwendungen während ihrer Ausführung lassen sich die kopierten Bereiche zweifelsfrei erkennen (Kapitel 3.1).

Vor allem der Einsatz der Attack Scripting Language (ASL), die durch Herrn Marc Ruef im Rahmen des ATK-Projekts entwickelt wurde und in verschiedenen Publikationen besprochen wird, kommt auch beim Interest Securitix Scanner zum Einsatz. Und auch wenn WEKA behauptet, dass den beiden Applikationen unterschiedliche Programmiersprachen zugrundeliegen und deshalb kein Code-Diebstahl gegeben sein kann, betrifft dies offensichtlicherweise die Plugins nicht. Diese sind nämlich bei beiden Produkten in der gleichen Programmiersprache (ASL) umgesetzt. (Kapitel 3.2)

Insgesamt weist das Exploit Repository des Interest Securitix Scanner eine Grösse von 101'023 Bytes auf. Dabei konnte festgestellt werden, dass 347 Plugins aus dem ATK übernommen wurden. Lediglich geringfügige Anpassungen wurden an einigen Stellen angestrebt, so dass eine unliebsame direkte Referenzierung auf die Original-Quelle nicht mehr so offensichtlich ist.

Doch selbst bei dieser amateurhaften „Bereinigung“, die nach dem zweiten Brief von Ruef scheinbar klammheimlich durch WEKA optimiert wurde [Ruef 2006b], konnten nicht alle offensichtlichen Querverweise entfernt werden. Die herauragendsten von diesen sind in Kapitel 3.2.1 aufgelistet. Dabei wurden grundsätzlich die folgenden Muster immerwieder ausgemacht:

- **Auftreten der Zeichenkette „ATK“:** Die Zeichenkette „ATK“ wird bei sehr vielen Plugins eingesetzt. Vor allem dann, wenn ein Platzhalter herbeigezogen werden musste. Dass auch der Interest Securitix Scanner den gleichen Platzhalter verwendet, der eine offensichtliche Referenzierung auf das ATK-Projekt darstellt, ist sehr auffällig. Diese vermeintliche Zufälligkeit konnte bei 7 Plugins zweifelsfrei ausgemacht werden.

- **Hinweise auf persönliche Daten von Marc Ruef:** In einigen Plugins werden als Platzhalter Zeitwerte eingesetzt. Dass per Zufall die gleichen von diesen herangezogen werden, erscheint ebenso unwahrscheinlich. Vor allem dort wo das Datum, welches eigentlich im Rahmen des Tests zufällig gewählrt werden kann, mit 11.02.81 ausgewiesen ist. Dies ist das Geburtsdatum von Ruef. Die Übernahme von Datumswerten mit typischer Charakteristik kann in 2 Plugins zweifelsfrei ausgemacht werden.

- **Übernahme von Fehlern:** Die Grösse des ATK-Projekts lässt es nicht verhindern, dass auch dort gewisse Fehler gemacht werden. Diese Fehler wurden 1 zu 1 übernommen und weisen damit klar die Kopie und damit die Richtigkeit der Vorwürfe aus. Diese herausragende Charakteristika kann in 5 Plugins nachgewiesen werden.

- **Fehlerhafte Anpassungen:** Bei der unrechtmässigten Anfertigung der Kopie wurden einige Anpassungen gemacht, um die Herkunft der Codeteile nicht so offensichtlich bestehen zu lassen. Diese Modifikationen führten dazu, dass die Funktionsweise der kopierten Funktionen untergraben und damit die betroffenen Plugins unbrauchbar werden. Es ist davon auszugehen, dass der Diebstahl durch eine Person erfolgte, die nicht genau wusste, was sie da überhaupt kopierte und anpasste. Dieses spezifische Verhalten kann bei 3 Plugins nachgewiesen werden.

Die in diesem Dokument vorgelegte Analyse lässt den Schluss zu, dass die durch Ruef sowie verschiedenen Medienberichterstatter (z.B. Heise) an die Öffentlichkeit getragenen Vorwürfen den Tatsachen entsprechen. Herr Ruef sieht sich aufgrund dessen nicht gezwungen, seine Aussagen zurückzuziehen oder in Zukunft keine Kommentare mehr zu den Gegebenheiten machen zu wollen.

2.2. Fragen und Antworten

F: Hat WEKA unrechtmässig Programmcode aus dem ATK Projekt in das Produkt Securitix übernommen?
A: Ja. (Kapitel 2.1)

F: Welche Teile sind von diesem Diebstahl betroffen?
A: Insgesamt bietet Securitix 437 Plugins aus dem ATK-Projekt an, ohne die Lizenzrechte (GPL) des Projekts und der Plugins und damit das Urheberrecht von Ruef anzuerkennen. (Kapitel 3.2.1)

F: Durch welche Mittel lässt sich dieser Diebstahl beweisen?
A: Am einfachsten ist die Gegenüberstellung der beiden Produkte. Es ist auf einen Blick zu sehen, dass die gleichen Plugins angeboten werden (Kapitel 3.1). Bei näheren Betrachtungen können gar in den übernommenen Teilen Hinweise auf die Original-Quelle gefunden werden. Die Zeichenkette „ATK“ und das Geburtsdatum von Ruef (11.02.81) tauchen immerwieder auf (Kapitel 3.2).

F: Kann dies nicht nur ein Zufall sein?
A: Nein. Die Plugins weisen eine gewisse Komplexität auf, die es mathematisch beweisbar macht, dass hier von keinem Zufall gesprochen werden kann. Die Namensgebung der Tests, deren Konstruktion, die zugrundeliegende Sprache, eingesetzten Muster und genutzten Zufallswerte können kein Zufall sein. (Kapitel 4)

F: Erreichen die Plugins die für eine juristische Streitigkeit erforderliche Schöpfungshöhe?
A: Ja. Die Plugins nutzen eine eigene Programmiersprache und stellen damit im Rahmen dieser spezifische Konstrukte zur Bewältigung von Aufgaben dar. Diese Aufgaben werden teilweise in besonderer, noch nie dagewesener Weise angegangen. (Kapitel 4.1)

F: Wer ist Urheber der betroffenen Plugins?
A: In allen Fällen ist dies Marc Ruef. Wurden Plugins im ATK-Projekt adaptiert (z.B. Inspiration durch Nessus NASL Plugins), ist dies unter Einhaltung der Lizenzrechte des Urhebers geschehen. Entsprechende Copyright-Hinweise (Quelle und Lizenz) finden sich ausdrücklich in sämtlichen Plugins im Feld plugin_created_name.

F: Sind abgesehen von den Plugins auch noch andere Codeteile betroffen?
A: Erste Abklärungen haben ergeben, dass scheinbar auch noch Erzeugnisse anderer Urheber übernommen wurden. Eine klare und definitive Äusserung hierzu ist aber zum gegenwärtigen Zeitpunkt nicht möglich.

3. Technische Analyse

3.1. Bildschirmüberprüfung

3.1.1. Ansicht der Plugins

Bevor eine umfassende technische Analyse durchgeführt werden soll (siehe Kapitel 3.2), soll eine Gegenüberstellung des Interest Securitix Scanner 1.6 sowie des Attack Tool Kit 4.1 erfolgen. Anhand dieser soll ohne technische Hilfe, sondern mit blossem Auge die kopierten Teile ausgemacht werden können.

In folgendem Screenshot ist das Exploiting-Modul von Securitix zu sehen (Abbildung 1). Wird dieses geöffnet, werden einem sämtliche Exploit-Routinen in einer alphabetischen Auflistung vorgestellt:

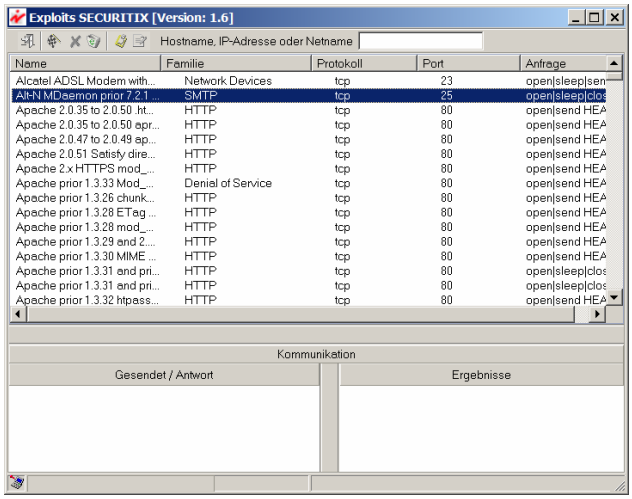


Abbildung 1: Die Exploiting-Route von Securitix

Ein Screenshot dieses Verhaltens war bis zur Schliessung der offiziellen Produktwebseite im Juni 2006 ebenso auf dieser zu sehen (Abbildung 2):

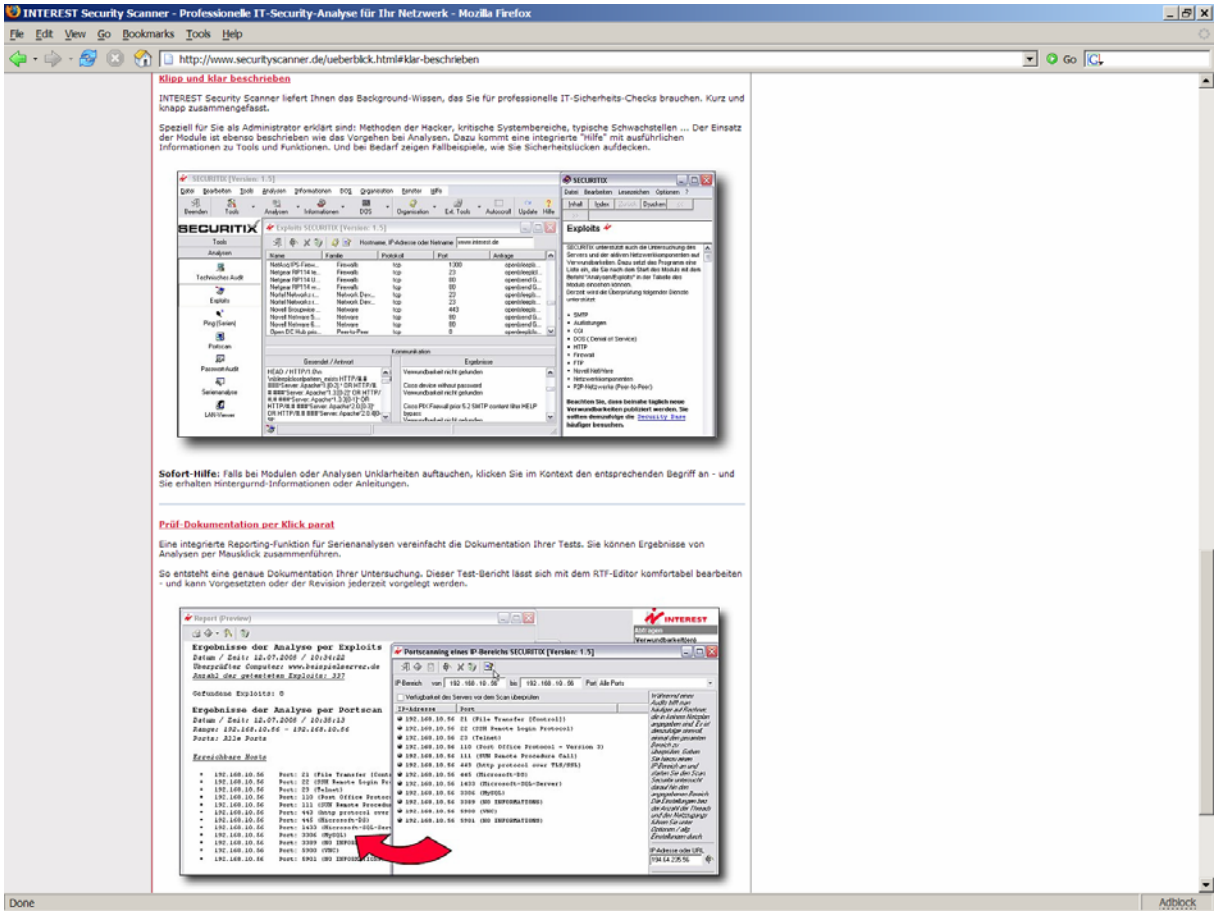


Abbildung 2: Offizieller Screenshot zum Exploiting von Securitix (Mai 2006)

Wird nun ebenfalls irgendeine offizielle Version des ATK gestartet – in folgendem Fall der neueste Release ATK 4.1 -, zeigen sich einem ebenfalls die gegebenen Plugins. Die Auflistung erfolgt wahlweise gleich in alphabetischer Reihenfolge. Dadurch ergibt sich grundsätzlich die identische Darstellung, wie sie auch bei Securitix übernommen wurde (Abbildung 3):

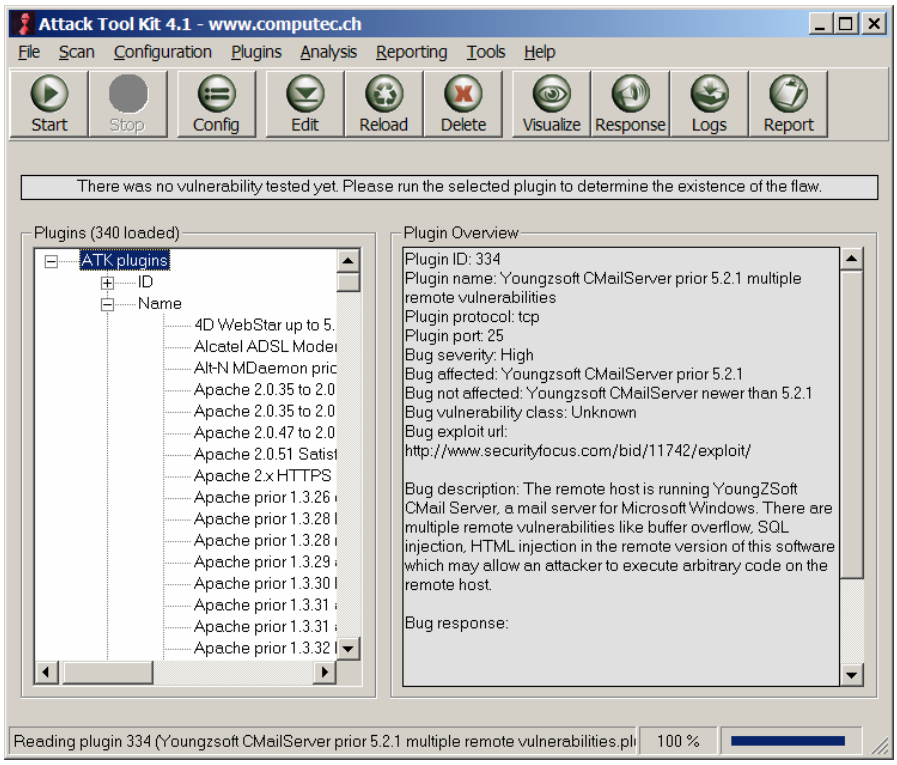


Abbildung 3: Die Auflistung der Original-Plugins

Anhand dieses optischen Vergleichs ist schon einmal auffällig, dass offensichtlich die gleichen Namen für die Tests verwendet werden. Einziger Unterschied in den gegebenen Screenshots ist, dass scheinbar das ATK Ziffern vor dem Alphabet listet und der Securitix Scanner nicht (siehe Plugin mit dem Namen „4D WebStar up to 5.3.3 symbolic link vulnerability“). Aber ab dem Plugin „Alcatel ADSL Modem without password“ werden die exakt gleichen Pluginnamen in exakt gleicher Reihenfolge aufgelistet.

3.1.2. Vergleich offensichtlicher Routinen

Securitix bietet in der gewährleisteten Darstellung die Möglichkeit, die Exploit-Routinen, die sich in Kapitel 4 als ASL-Konstrukte erweisen, einzusehen. Somit ist in der Spalte „Anfrage“ ersichtlich, welche Kommandos die Software zu nutzen pflegt, um eine Sicherheitsüberprüfung für eine dedizierte Schwachstelle durchzuführen (Abbildung 4):

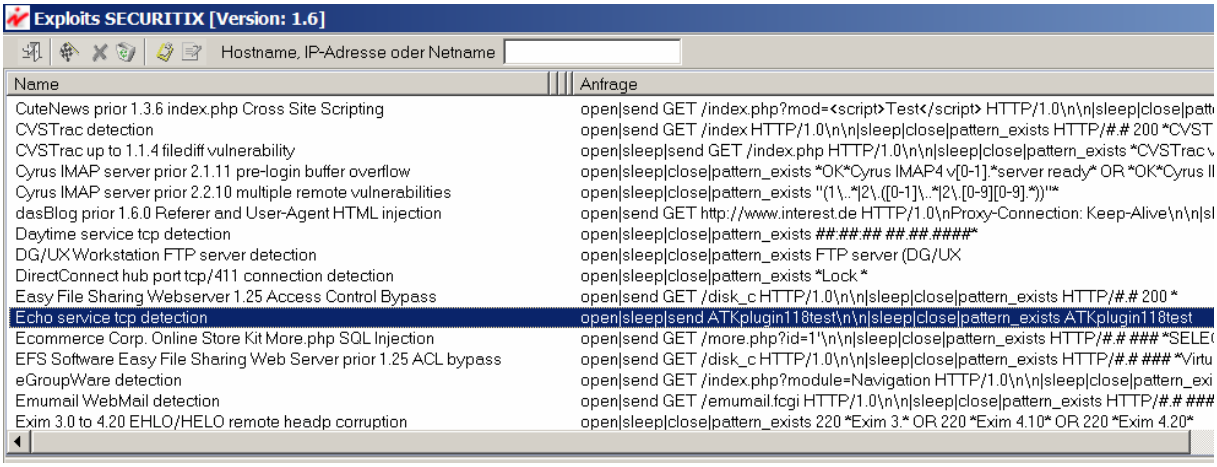


Abbildung 4: Ein offensichtlich übernommenes Plugin (ATKplugin118test)

Bei der Betrachtung der jeweiligen Routinen fallen einige Dinge auf. So finden sich beispielsweise an verschiedenen Stellen die Zeichenketten „ATK“, anderweitige Hinweise auf das ATK-Projekt oder sogar dessen Entwickler Marc Ruef. In Abbildung 4 ist beispielsweise das Plugin „Echo service tcp detection“ angewählt. In diesem wird die Zeichenkette „ATKplugin118test“ genutzt. Eine klare Referenz zum ATK-Projekt, denn innerhalb dieses weist das ATK-Plugin die ID 118 auf.

In der folgenden Tabelle werden die offensichtlichsten Plugins, die bei der Übernahme nicht oder nur mangelhaft kaschiert wurden, aufgelistet. Diese stellen damit in keinster Weise eine komplette Liste der übernommenen Plugins dar – Insgesamt wurden nämlich ganze 347 angeboten. Viel mehr ist es nur eine gefilterte Liste, anhand derer die Übernahme mittels verschiedener Indizen zweifelsfrei und ohne grosse Aufwände belegt werden kann.

ID	Name	Beweis
13	Daytime service tcp detection	Das Pattern zur Verifikation der Schwachstelle lautet ##:##:## ##.##.####* und kann damit leider nur Windows-Implementierungen des Daytime-Dienstes erkennen. Die gleiche Fehlbarkeit wurde damit auch bei Securitix begangen.
35	Proxy gopher support detection	Ursprünglich erfolgte der Zugriff auf den TCP-Port 1234 am System www.computec.ch. Dieser Port war, bis zum letzten Server-Umzug (von Exigo zu Metanet), auch belegt. Das Securitix-Plugin wurde einfach auf www.interest.de geändert. Da dort kein Dienst angeboten wird, kann das Plugin gar nicht funktionieren. Derjenige, der es kopiert hat, hat dessen Sinn und Zweck nicht verstanden.
46	Macintosh OS X FTP server detection	Das Pattern zur Verifikation des erfolgreichen Zugriffs weist zwischen „Version:“ und „Mac“ eine Leerstelle zuviel auf. Auch dieser unnötige und damit das Plugin nicht richtig nutzbare Fehler wurde übernommen.
90	phpBB viewtopic.php cross site scripting	Der Testzugriff wird auf das Forum 10 ausgeführt. Die Zahl ist jedoch nicht für den Test relevant. Ebenso die Anzahl Tage, die angezeigt werden soll. Sie lautet 99. Jede andere Zahl hätte hier eingesetzt werden können, stattdessen wurden die gleichen übernommen.

91	phpBB viewforum.php cross site scripting	⇒ siehe Plugin ID 90
97	Oracle 9iAS iSQLplus cross site scripting	Das Plugin zieht den fiktiven Benutzer joe heran. Und dies, obschon die Wahl des Benutzernamens beim Test grundsätzlich egal ist.
95	Moodle up to 1.4 post.php cross site scripting	Das Plugin verschickt die Zeichenkette „ATK plugin to detect post.php flaw“. Dies ist ein eindeutiger Hinweis auf das ATK-Projekt. Jede andere Zeichenkette hätte auch zum Erfolg geführt.
111	Microsoft Frontpage detection	Der Zugriff erfolgt mit dem fiktiven Datum „Mon, 23 Mar 2003 01:23:42 GMT“. Es wurde das gleiche Datum übernommen. Auch der Fehler, dass der 23. März 2003 gar kein Montag, sondern ein Sonntag war.
118	Echo service tcp detection	Es wird die Zeichenkette ATKplugin118test überprüft. Dies ist ein klarer Hinweis auf das ATK sowie das gleichnamige Plugin mit der internen ID 118.
156	SMTP mail relay test	Im Rahmen des Mailtests wird eine SMTP-Verbindung realisiert. In dieser wird sich mit atk.test angemeldet, als Absender die Adresse plugin156@atk.test und als Datum 11. Feb 81 um 11:02:11 gewählt. Erstere sind eine Referenz auf das ATK-Projekt bzw. das gegebene Plugin mit der ID 156. Letzteres ist das Geburtsdatum von Marc Ruef (11.02.1981).
165	SMTP spam filter test	⇒ siehe Plugin ID 156
187	PsNews prior 1.2 index.php function Cross Site Scripting	Für Testzwecke wird die Zeichenkette „atk 253“ verschickt. Dies ist eine eindeutige Referenz auf das entsprechende ATK-Plugin. In der übernommenen Version wurde der Pattern-Zugriff einfach auf „test“ geändert. Das Plugin kann damit gar nicht mehr funktionieren, da der Reiz und die Reaktion nicht mehr identisch sind. Die Funktionsweise des Plugins wurde nicht verstanden.
190	Keene Digital Media Server prior 1.0.3 slideshow.kspx Cross Site Scripting	Aufgrund eines Fehlers schickt das Original-Plugin des ATK eine Anfrage für „foo“, überprüft aber die Zeichenkette „atk“. Der Test kann so nicht funktionieren. Der gleiche Fehler, der auf das ATK referenziert, ist auch beim übernommenen Plugin gegeben.
194	Jerod Moemeka Xedus detection	Der für Testzwecke übermittelte Parameter lautet „Attack Tool Kit“. Dies ist ein eindeutiger Hinweis auf das ATK-Projekt.
198	Symantec Raptor Firewall 6.5 Simple Secure Web Server detection	Für den Testzugriff wird die Zeichenkette „ATK 198 test request“ verschickt. Dies ist eine Referenzierung auf das ATK-Projekt bzw. das entsprechende Plugin mit der ID 198.
199	Symantec Raptor Firewall 6.5 weak ISN detection	⇒ siehe Plugin ID 198
220	Pinnacle ShowCenter BSE web server skin denial of service	Das Plugin referenziert auf ein nicht-existentes Skin mit dem Namen „ATK“. Eine eindeutige Referenzierung auf das ATK-Projekt. An dieser Stelle hätte jeder beliebige Wert eingesetzt werden können, wie Ruef in der Veröffentlichung seines Advsories bekanntgab [Ruef 2004c].
232	BBS E-Market Professional Arbitrary File Inclusion	Es wird der Test mit der URL http://www.php.net/downloads.php durchgeführt. Dieser kann aber mit jeder beliebigen URL umgesetzt werden.
297	ftpd cwd user detection	Als Passwort für den FTP-Zugriff wird atk@test.example verwendet. Dies ist ein eindeutiger Hinweis auf das ATK-Projekt. Siehe auch Plugin ID 156 und 165.

Tabelle 2: Die am einfachsten beweisbaren Übernahmen

Besonders offensichtlich sind die Übernahmen dort, wo eine direkte Referenzierung auf das ATK-Projekt stattfindet. Vorzugsweise die Zeichenkette „ATK“ spielt bei einer Vielzahl der Plugins eine zentrale Rolle. Diese wird immer dann eingesetzt, wenn ein Platzhalter herbeigezogen werden soll. Also eine frei wählbare Zeichenkette.

Die Analyse dieser Art kann sehr einfach mit der folgenden Eingabe durchgeführt werden. Hierbei wird auf einem Microsoft Windows XP der Inhalt der Datei ex.tr0 durch das Kommando type eingelesen und auf dem Bildschirm durch das Kommando find ausschliesslich jene Zeilen ausgegeben, in denen die Zeichenkette „ATK“ vorkommt. Durch den Schalter /C wird lediglich die Anzahl der übereinstimmenden Zeilen ausgewiesen, wobei mit /I die Gross-/Kleinschreibung des Suchmusters ausser Acht gelassen wird. Hierbei sind 7 Routinen mit der Grossschreibung „ATK“ und 5 mit der Kleinschreibung „atk“ als solche identifiziert (insgesamt 12):

```
01   C:\weka-atk\type ex.tr0 | find /C /I "ATK"
02   12
```

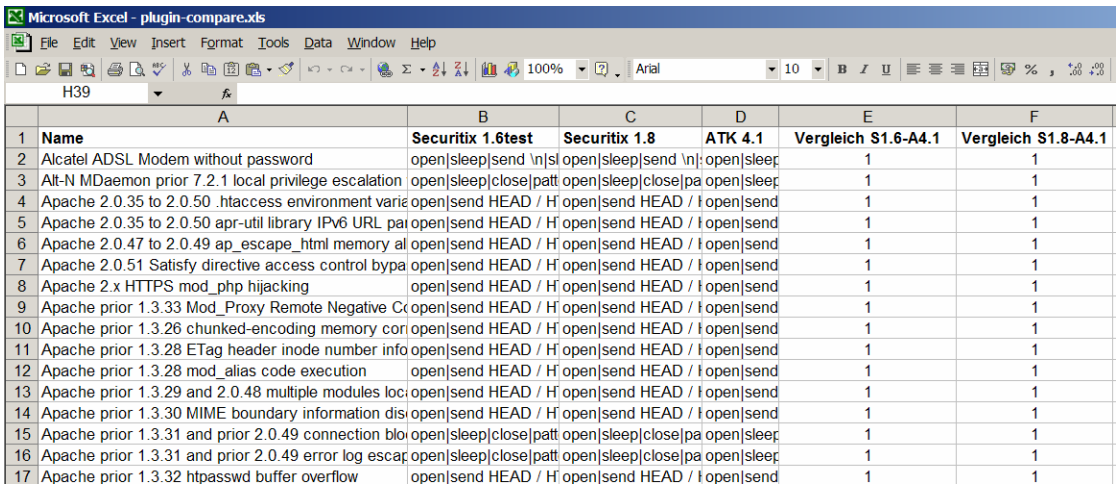
Aber auch an anderen Stellen, wo Werte übernommen wurden, die eigentlich für den Testzugriff zufällig gewählt werden hätten können, ist die Kopie ersichtlich.

3.1.3. Vergleich der 1 zu 1 übernommenen Plugins

Es gibt eine Vielzahl unterschiedlicher formaler Algorithmen, anhand derer der Unterschied (Delta) zwischen zwei Zeichenketten berechnet werden kann. Um es in einer ersten Phase möglichst einfach zu halten, wird ein Direktvergleich zweier Zeichenketten S_1 und S_2 vorgenommen. Ist die Equivalenz der Form $S_1 = S_2$ gegeben, wird das Plugin als 1 zu 1 kopiert identifiziert. Ist auch nur eine Abweichung gegeben (ein Zeichen verändert oder Gross-/Kleinschreibung angepasst), ist keine 1 zu 1 Übernahme identifiziert.

Zur Beweisführung liegt eine umfassende Tabelle vor, die zeigt, welche Plugins 1 zu 1 (ohne jegliche Änderung) übernommen wurden. Es ist dabei der Unterschied zwischen den Versionen 1.6 und 1.8 von Securitix auszumachen. Interessanterweise wurden eben jene Dinge, die von Ruef in seinem zweiten eingeschriebenen Brief als eindeutig übernommen identifiziert wurden, angepasst.

Diese tabellarische Auflistung ist für die ersten Datensätze in Abbildung 5 festgehalten. In der Spalte A wird der Name des Plugins, der sowohl bei Securitix als auch ATK identisch ist, angegeben. In der Spalte B wird der ASL-Code, wie er sich in Securitix 1.6test findet und in Spalte C wie er sich in 1.8 findet dargestellt. Der Original-Code, wie er in ATK 4.1 genutzt wird, ist in Spalte D abgebildet. In der Spalte E wird der Direktvergleich zwischen Securitix 1.6test und ATK 4.1 und in Spalte F der Vergleich zwischen Securitix 1.8 und ATK 4.1 vorgenommen. Kann eine 1 zu 1 Übereinstimmung ausgemacht werden – hierzu wird die Formel IF($S_1 = S_2$;1) verwendet -, so wird die Ziffer 1 abgebildet. Andernfalls der Begriff FALSE. Damit wird für die jeweiligen Spalten eine Addition mittels SUM() möglich.


Microsoft Excel - plugin-compare.xls

File Edit View Insert Format Tools Data Window Help

H39

	A	B	C	D	E	F
1	**Name**	**Securitix 1.6test**	**Securitix 1.8**	**ATK 4.1**	**Vergleich S1.6-A4.1**	**Vergleich S1.8-A4.1**
2	Alcatel ADSL Modem without password	open|sleep|send \n|s	open|sleep|send \n|	open|sleep	1	1
3	Alt-N MDaemon prior 7.2.1 local privilege escalation	open|sleep|close|patt	open|sleep|close|pa	open|sleep	1	1
4	Apache 2.0.35 to 2.0.50 .htaccess environment varia	open|send HEAD / H	open|send HEAD / H	open|send	1	1
5	Apache 2.0.35 to 2.0.50 apr-util library IPv6 URL pa	open|send HEAD / H	open|send HEAD / H	open|send	1	1
6	Apache 2.0.47 to 2.0.49 ap_escape_html memory al	open|send HEAD / H	open|send HEAD / H	open|send	1	1
7	Apache 2.0.51 Satisfy directive access control bypa	open|send HEAD / H	open|send HEAD / H	open|send	1	1
8	Apache 2.x HTTPS mod_php hijacking	open|send HEAD / H	open|send HEAD / H	open|send	1	1
9	Apache prior 1.3.33 Mod_Proxy Remote Negative C	open|send HEAD / H	open|send HEAD / H	open|send	1	1
10	Apache prior 1.3.26 chunked-encoding memory cor	open|send HEAD / H	open|send HEAD / H	open|send	1	1
11	Apache prior 1.3.28 ETag header inode number info	open|send HEAD / H	open|send HEAD / H	open|send	1	1
12	Apache prior 1.3.28 mod_alias code execution	open|send HEAD / H	open|send HEAD / H	open|send	1	1
13	Apache prior 1.3.29 and 2.0.48 multiple modules loc	open|send HEAD / H	open|send HEAD / H	open|send	1	1
14	Apache prior 1.3.30 MIME boundary information dis	open|send HEAD / H	open|send HEAD / H	open|send	1	1
15	Apache prior 1.3.31 and prior 2.0.49 connection blo	open|sleep|close|patt	open|sleep|close|pa	open|sleep	1	1
16	Apache prior 1.3.31 and prior 2.0.49 error log escap	open|sleep|close|patt	open|sleep|close|pa	open|sleep	1	1
17	Apache prior 1.3.32 htpasswd buffer overflow	open|send HEAD / H	open|send HEAD / H	open|send	1	1

Abbildung 5: Vergleich der Plugins

Bei diesem grundsätzlich einfachen Vergleich (ohne komplexe Algorithmen) wird aufgezeigt, dass insgesamt 337 ATK-Plugins übernommen wurden. In der ersten Version, die Ruef zugänglich war (Securitix 1.6test), konnten 300 Plugins als 1 zu 1 Kopie identifiziert werden. Dies entspricht 89,02 %. Nach den Anpassungen in der neueren Version (Securitix 1.8) waren dies nur noch 290 übernommene Plugins. Dies entspricht 86.05 %. Die Abnahme der Übernahmen ist damit durch 2.97 % gegeben. Eine marginale Differenz, die in Anbetracht der Vielzahl der nachweislich übernommenen Codeteile keinen Unterschied macht.

3.2. Reverse Engineering

3.2.1. Identifikation der Plugin Repositories

Sämtliche Plugins des ATK werden im statischen Unterverzeichnis \plugin bereitgestellt. In diesem lokalen Repository ist jeder einzelne Check durch eine dedizierte Datei mit der Endung .plugin gewährleistet. Der Name der jeweiligen Plugins orientiert sich an deren Titel. So ist auf einen Blick ersichtlich, zu welchem Zweck welche Plugin-Datei eingesetzt werden kann (Abbildung 6):

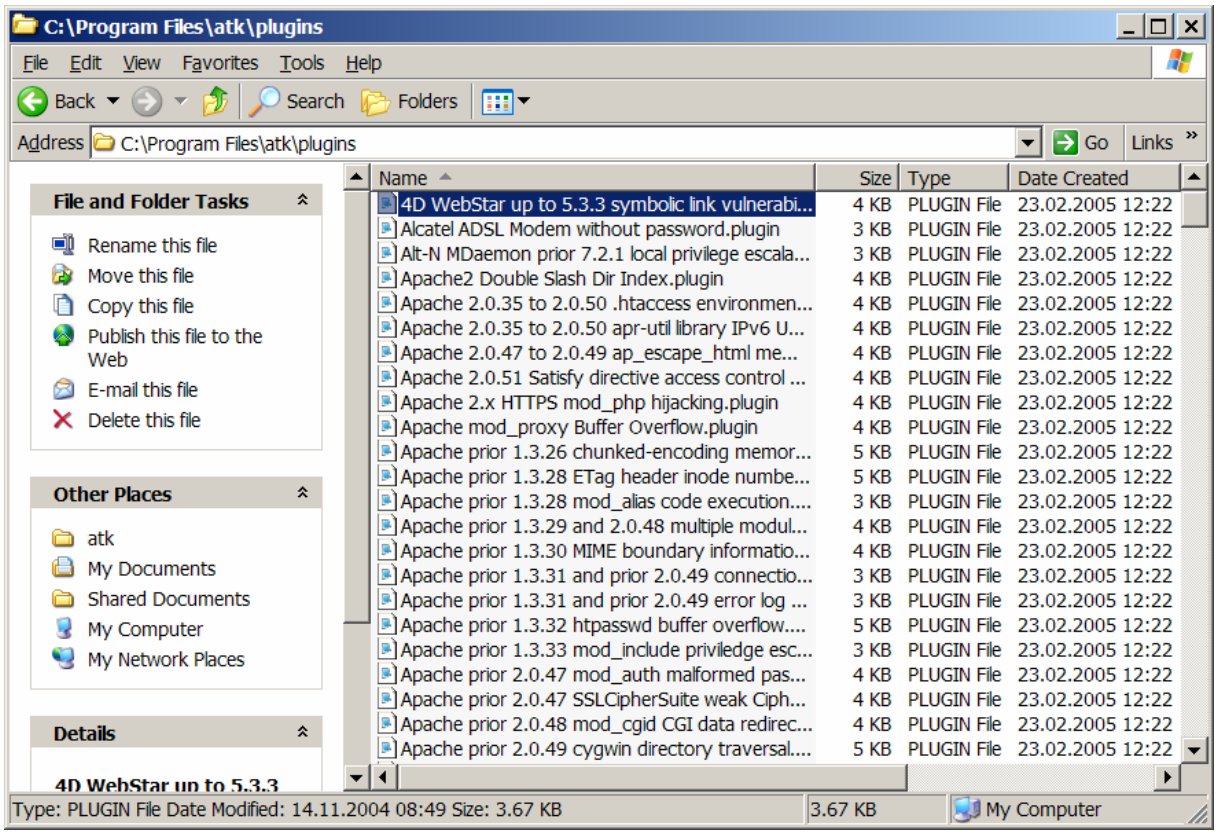


Abbildung 6: Auflistung der Original-Plugins im Plugins-Ordner

Im Gegensatz verwendet der Interest Securitix Scanner 1.x (sowohl 1.6 als auch 1.8) eine Datei mit dem Namen ex.tr0, um die übernommenen Plugins bereitzustellen (Abbildung 7):

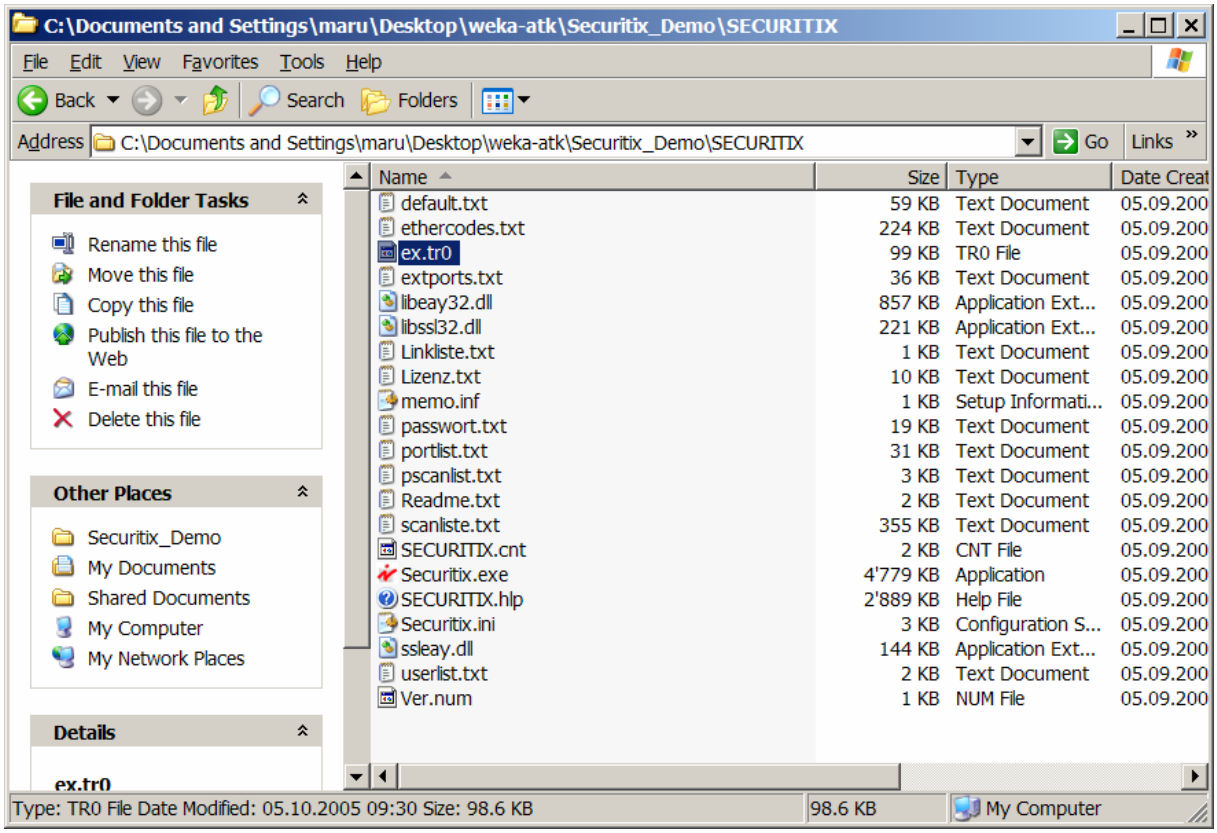


Abbildung 7: Die zusammenkopierte Exploit-Datei

Die Einsicht dieser Datei ist ohne grössere Umschweife und das Heranziehen komplexer Reverse Engineering-Software (z.B. Hex-Editor oder Disassembler) möglich. Durch ein einfaches Umbenennen der Datei von ex.tr0 zu ex.txt kann von der automatischen Dateierweiterungs-Assoziation von Microsoft Windows Gebrauch gemacht werden. Sodann lässt sich die gegebene Datei mittels einem Doppelklick in der dafür vorgesehenen Textverarbeitungs-Software öffnen. Die übernommenen Plugins und die einzelnen ASL-Strukturen sind damit, im Beispiel wird WordPad genutzt, ersichtlich (Abbildung 8):

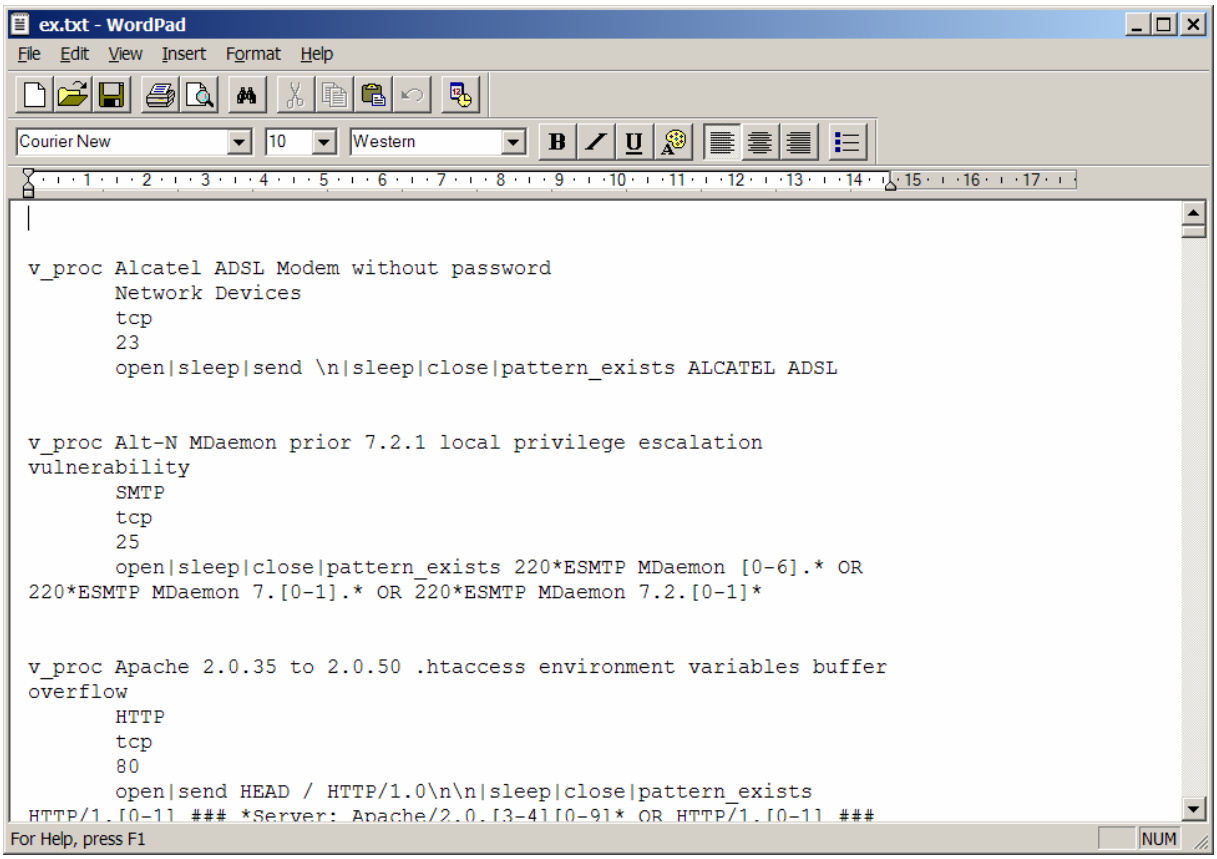


Abbildung 8: Der Inhalt der adaptierten Exploit-Datei

Eine Analyse der Besonderheiten, die nicht nur die Adaption der ASL-Struktur sondern auch gleich die komplette Übernahme der Routinen beweist, ist in Kapitel 4 zusammengefasst.

4. Formale Beweisführung

4.1. Hintergrund der Attack Scripting Language

WEKA weist in ihren Stellungnahmen, sowohl persönlich an Marc Ruef als auch jene an die Medien, immerwieder daraufhin, dass keine Codeteile übernommen wurden. Als Argument wird angeführt, dass das ATK in Microsoft Visual Basic 6.0 und Securitix in Delphi geschrieben wurde. Der Vorwürf der unrechtmässig übernommenen Teile gründet sich jedoch auf den Plugins, die in einer eigenen Programmierpsrache geschrieben werden. Wie die nachfolgenden Ausführungen zeigen werden, ist damit der vorgeworfene Code-Diebstahl gegeben.

Die Bestrebungen von Ruef waren, eine Software bereitzustellen, die Sicherheitsüberprüfungen automatisieren können sollen. Da tagtäglich neue Schwachstellen bekannt werden, sollte ein Plugin-basierter Ansatz gewählt werden. Die einzelnen Tests werden durch dedizierte Dateien, sogenannte Plugins, gewährleistet. Werden neue Schwachstellen bekannt und diese in den Plugins umgesetzt, müssten die Benutzer des ATK nur die neuen Plugins und nicht das gesamte Software-Paket herunterladen. Zu diesem Zweck wurde innerhalb des ATK ab Version 4.0 ein AutoUpdate-Feature, das diesen Download automatisiert, integriert.

Ruef entwickelte für das ATK Projekt bzw. die genutzten Plugins eine eigene Programmiersprache mit dem Namen ASL (Attack Scripting Language). Diese wurde zeitgleich mit der Herausgabe der ersten Version des ATK im Jahr 2003 der Öffentlichkeit vorgestellt [Ruef 2003]. Das Ziel von ASL war, eine möglichst einfache und dynamische Sprache bereitzustellen. Es sollte möglich sein, dass auch während einer Sicherheitsüberprüfung manuell und ohne grössere Aufwände Anpassungen an den Plugins vorgenommen werden können. Dies hebt ASL eindeutig von den Implementierungen der Python Skripte bei MetaSploit (http://www.metasploit.com) oder den NASL-Plugins des Nessus Projekts (http://www.nessus.com) ab. Die Implementierung von ASL stellt damit einen Grundpfeiler für den Erfolg des ATK dar, wie dies in einer Präsentation von Marc Ruef (Seite 8) festgehalten wurde [Ruef 2004b][1]:

> Was macht das ATK besser? Erstellen und anpassen von Checks ohne Programmierkenntnisse möglich.

ASL-Plugins weisen seit ATK 0.2 (September 2003) eine XML-ähnliche Struktur auf. In dieser werden einzelne Tags eingesetzt, dank denen die jeweiligen Datensätze der Plugins referenziert werden können [Ruef 2004c]. Das Herzstück der Plugins ist die Detection bzw. Exploit Routine. Durch diese wird die eigentliche Sicherheitsüberprüfung gewährleistet. Dabei kommt eine Kommandostruktur der folgenden Form zum Einsatz (Beispiel):

```
open|send GET / HTTP/1.0\n\n|sleep|close|pattern_exists HTTP/1.# ###
```

Eine alternative Darstellung (ListView), die mit ATK 4.0 eingeführt wurde, lässt das gleiche Beispiel-Plugin der folgenden Form darstellen:

```
01   open|
02   send GET / HTTP/1.0\n\n|
03   sleep|
04   close|
05   pattern_exists HTTP/1.# ###
```

Diese Sprache weist einige Besonderheiten auf, die sie massgeblich von anderen Programmiersprachen unterscheidet. Auf diese Beonderheiten, die nicht durch Zufall oder Versehen bei Interest Securitix Scanner ebenfalls gegeben sein können, soll nachfolgend eingegangen werden.

[1] Gleichzeitig ist die Einfachheit von ASL in gewissen Situationen hinderlich. Denn damit können keine vergleichbar komplexen Plugins wie bei NASL oder Python realisiert werden. Erweiterte Interne Funktionen (z.B. arithmetische Operatoren) fehlen bisweilen.

4.2. Delimiter der Kommandos

Die moderne Programmierung sieht vor, dass durch eine Aneinanderreihung von Befehlen eine Automatisierung einer Funktion gewährleistet wird. Dabei ist es erforderlich, dass im Kontext der Programmiersprache die einzelnen Anweisungen klar voneinander getrennt werden. Der klassische Ansatz, der bei vielen populären Programmiersprachen zum Tragen kommt (z.B. C, Java und PHP) , ist durch den Strichpunkt gegeben. Nach jedem eigenständigen Befehl hat ein solcher zu folgen. Nachfolgend ein Beispiel einer einfachen Funktion in ANSI C [WikiPedia 2006]. Die jeweiligen Trenzzeichen der dedizierten Befehle sind farblich hervorgehoben (Zeilen 06, 07 und 10):

```
01   #include <stdio.h>
02
03   int main(int argc, char **argv)
04   {
05       if (printf("Hallo Welt!\n") < 0) {
06           fprintf(stderr, "Error occured while printing to stdout\n");
07           return 1;
08       }
09
10       return 0;
11   }
```

Derartige Konstrukte sind, so war Ruef bei der Entwicklung von ASL der Meinung, zu komplex für den dynamischen Gebrauch. Aus diesem Grund werden bei ASL die jeweiligen Konstrukte auf einer Zeile geschrieben und die Kommandos durch das Pipe-Zeichen (|) voneinander getrennt. Dies ist eine Besonderheit, die, soweit bekannt, nur von ASL umgesetzt wird[2]. Sie ist an nachfolgendem Beispiel wiederum hervorgehoben:

```
open|send GET / HTTP/1.0\n\n|sleep|close|pattern_exists HTTP/1.# ###
```

Bei der Gegenüberstellung von Securitix zu ATK kann mit blossem Auge ausgemacht werden, dass auf ASL zurückgegriffen wird. Es werden nämlich ebenso die jeweiligen Funktionen durch das Pipe-Zeichen, und nicht wie sonst der übliche Strichpunkt, voneinander getrennt. Ebenso, wie Kapitel 4.3 zeigen wird, kommen die gleichen Funktionen und Syntax' zum Einsatz.

[2] Viele Programmiersprachen pflegen dem Zeichen | eine besondere Funktion beizumessen. Entweder wird es als Pipe verstanden, die zur Weiterleitung der Ausgabe eines Programms als Eingabe eines anderen Programms genutzt wird. Populäre Beispiele sind die jeweiligen Shell-Implementierungen (z.B. csh, bash). Oder es übernimmt die Funktion der logischen ODER-Verknüpfung (||). Populäres Beispiel hierzu ist PHP.

4.3. Genutzter Befehlssatz

Höhere Programmiersprachen kommen meist mit einem erweiterten Befehlssatz und zusätzlichen Funktionen daher, dank denen gewisse Implementierungen besonders einfach angestrebt werden können. ASL setzt seit jeher die folgenden Befehle ein [Ruef 2003]:

Befehl	Beschreibung
open	Öffnet eine Socket-Verbindung zum Zielsystem.
sleep [sekunden]	Wartet eine gewisse Anzahl Sekunden, bevor das nächste Kommando ausgeführt wird.
send <daten>	Schickt die definierte Zeichenkette über die etablierte Verbindung. Dabei wird das Sonderzeichen \n vom ATK als Newline-Zeichen interpretiert.
Close	Schliesst die offene Socket-Verbindung zum Zielsystem.
pattern_exists <pattern>	Überprüft, ob in einer Ausgabe die definierte Zeichenkette vorhanden ist. Falls dem so ist, wird die überprüfte Schwachstelle als gegeben ausgewiesen.
pattern_not_exists <pattern>	Die Umkehrfunktion von pattern_exists. Es wird im Gegensatz überprüft, ob eine Zeichenkette nicht existiert. Ist dem so, wird die Schwachstelle als gegeben ausgewiesen.
icmp_alive	Durch den Versand einer ICMP echo request-Anfrage kann manuell die Erreichbarkeit eines Systems verifiziert werden. Dies ist besonders bei der Überprüfung von Denial of Service-Attacken erforderlich.
run <kommando>	Durch den run-Befehl, dieser wird bisher in keinen offiziellen Plugins eingesetzt, lassen sich lokale Kommandos ausführen. Damit können auch weitere Tools oder gar Exploits eingebunden werden.

Tabelle 3: Der Grundlegende Befehlssatz von ASL

Betrachtet man andere Programmiersprachen und Scanning-Produkte, so fällt auf, dass grundsätzlich komplett andere Befehlssätze zum Einsatz kommen. Zum Erstaunen wird jedoch auch bei Securitix entgegen der Zurückweisung der Vorwürfe der gleiche Befehlssatz eingesetzt. Es wurden keine neuen Befehle eingeführt und ebenso auch nicht auf die Implementation einzelner Funktionen verzichtet. Ebenso ist der Syntax der Funktionen, wie er in der Tabelle 3 aufgelistet wird, gleich geblieben.

4.4. Reguläre Ausdrücke der pattern-Funktionen

Wie in Absatz 4.3 dargelegt ist, wurden mit ASL einige individuelle Funktionen erstellt, die ebenso bei Securitix adaptiert wurden. Dabei weisen die beiden Funktionen pattern_exists und pattern_not_exists eine besondere Charakteristik auf. Diese lassen als Parameter eine Zeichenkette, die überprüft werden soll, zu. Beispielsweise überprüft der Befehl „pattern_exists ATK“, ob in der Rückgabe des getesteten Systems in irgendeiner Weise die Zeichenkette „ATK“ vorkommt.

Da eine solch starre Validierung nicht zeitgemäss ist, bietet ASL seit jeher das Nutzen von regulären Ausdrücken an. Diese erlauben die formale Definition einer Maske, anhand derer das Auftreten komplexer Zeichenkombinationen überprüft werden kann. Hierbei kommen verschiedene Sonderzeichen zum Einsatz, wie Tabelle 4 zeigt:

	Sonderzeichen	Beschreibung
A	#	An dieser Stelle kann eine Ziffer [0-9] gegeben sein.
B	[3-7]	An dieser Stelle können die Ziffern {3, 4, ... 7} stehen
C	*	An dieser Stelle können 0 oder mehr Zeichen stehen.
D	OR	Eine logische ODER-Verknüpfung zweier Patterns erlaubt das Erstellen komplexer Suchmasken. Beispiel „ATK #.# OR Attack Tool Kit #.#“.

Tabelle 4: Die speziellen regulären Ausdrücke von VB6 Like

Die Sonderzeichen {A, B, C} werden durch eine interne Funktion von der Programmiersprache Microsoft Visual Basic 6.0, in der das ATK bis Version 4.1 geschrieben wurde, bereitgestellt. Diese Funktion lautet „Like“ und stellt eine sehr spezielle Form der Implementierung einer Überprüfung von regulären Ausdrücken dar. So unterscheiden sich reguläre Ausdrücke je nach Sprache in gewissen Punkten. Da das ATK die Like-Funktion von VB6 benutzt, werden in den ASL-Plugins auch eben jene regulären Ausdrücke verwendet. Von einer eigenen Implementierung eines Moduls zur Unterstützung mächtiger regulärer Ausdrücke wurde bisher abgesehen.

WEKA behauptet, dass Securitix in Delphi geschrieben wurde und deshalb gar keine Code-Übernahme mögich war. Es erscheint aber unsinnig, dass WEKA die reguläre Ausdrucksform, wie sie nur durch VB6 bei den ASL-Scripten durchgesetzt wurde, einsetzt. Es gibt eine Vielzahl besserer Implementierungen von regulären Ausdrücken – besonders auch bei Delphi -, so dass die Kompatibilität zur Like-Funktion von VB6 als nicht gerechtfertigt erscheint. Jedoch nur solange, bis man sich natürlich darum bemüht, sämtliche Plugins ohne grosse Anpassung übernehmen zu wollen.

5. Literaturverzeichnis

[Ruef 2003] Ruef, Marc, November 2003, How to write a plugin manually, ATK Project, http://www.computec.ch/projekte/atk/plugins/write_a_plugin_manually/

> Dies ist die offizielle Anleitung dazu, wie manuell eigene ATK Plugins geschrieben werden können. Hierbei wird auf die Struktur der Sprache sowie die Funktionsweise der einzelnen Befehle eingegangen. Die Charakteristika von ASL ist damit sehr einfach ersichtlich.

[Ruef 2004a] Ruef, Marc, 21. September 2004, Pinnacle ShowCenter Skin Denial of Service, Bugtraq Mailinglist, http://www.securityfocus.com/archive/1/375995

> Ruef entdeckte Mitte 2004 einen Fehler im Produkt Pinnacle ShowCenter. Diesen machte er auf verschiedenen Sicherheits-Mailinglisten publik. In seinem Advisory wies er darauf hin, dass eine die Wahl eines nicht-existenten Skins zu einer Denial of Service führen könne. Dabei sei die Wahl dessen ausschliesslich an dessen Nichtexistenz gebunden. Das dazugehörige ATK-Plugin referenzirt dabei auf das Skin mit dem Namen „ATK“. Dieses wurde bei Securitix übernommen.

[Ruef 2004b] Ruef, Marc, 28. September 2004, Attack Tool Kit (ATK) Project Presentation, http://www.computec.ch/projekte/atk/documentation/presentation-2004/atk-presentation-2004.ppt

> Diese Präsentation, die erstmals am SIAP Circle September 2004 in Zürich gehalten wurde, führt in die Idee des ATK-Projekts ein. Dabei wird unter anderem darauf hingewiesen, dass mit ASL absichtlich eine möglichst einfache Sprache geschaffte werden sollte, durch die sehr einfach und dynamisch noch während der Nutzung die erforderlichen Anpassungen gemacht werden können (Live-Editing).

[Ruef 2004c] Ruef, Marc, 13. Oktober 2004, A brief history of the ATK, http://www.computec.ch/projekte/atk/documentation/brief_history_of_the_atk/

> Durch diese Publikation wurde dem interessierten Nutzer des ATK dessen Geschichte und Entwicklung vorgetragen. So ist darin zu erkennen, dass eine erste Implementierung des ATK im August 2003 umgesetzt wurde. Schon damals kann eine sehr frühe Version der Attack Scripting Language (ASL) zum Einsatz. Das Grundgerüst der Prozeduren ist bis heute gleich geblieben.

[Ruef 2006a] Ruef, Marc, 22. Juni 2006, ATK Project gegen WEKA Business Information GmbH & Co. KG., Marcs Blog, computec.ch, http://www.computec.ch/news.php?item.117

> Dieser Blog Eintrag stellt die erste Veröffentlichung der Umstände dar. In diesem wird das bisherige Vorgehen beider Parteien bis zum Veröffentlichungsdatum dokumentiert. Daraus ist ersichtlich, dass Herr Ruef durch eine Vielzahl an eingeschriebenen Briefen, Emails und Telefongesprächen darum bemüht war, die Sache aussergerichtlich zu lösen. Dabei wurde ausschliesslich die Forderung gestellt, die Grundlagen der GPL, unter der das Attack Tool Kit veröffentlicht wird, einzuhalten. WEKA hat keine Anstalten gemacht, die Lizenz und damit das Urheberrecht von Herrn Ruef anzuerkennen.

[Ruef 2006b] Ruef, Marc, 18. Juli 2006, ATK gegen WEKA, Teil 2: Rückzug?, Marcs Blog, computec.ch, http://www.computec.ch/news.php?item.120

> In einem weiteren Blog Eintrag berichtet Ruef davon, dass in einer neueren Version der Software jene Teile, die in einem früheren Gespräch mit WEKA als eindeutig übernommen deklariert wurden, kommentarlos abgeändert wurden. So hätte die Referenzierung auf die Original-Quelle verschleiert werden sollen.

[WikiPedia 2006] WikiPedia Deutschland, 05. September 2006, C (Programmiersprache), http://de.wikipedia.org/wiki/C_(Programmiersprache)

> Die freie Enzykloädie WikiPedia stellt eine Vielzahl an Artikeln zu unterschiedlichen Themengebieten bereit. Unter anderem ist dort auch eine kleine Einführung in die Funktionsweise der Programmiersprache C zu finden. Das gegebene Hallo Welt-Beispiel (Absatz 4) wird in dieser Analyse zitiert.

6. Abbildungsverzeichnis

7. Tabellenverzeichnis