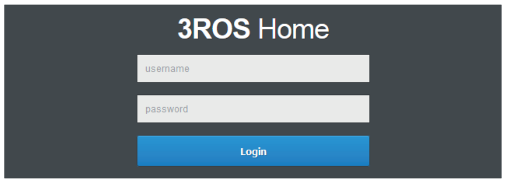
virus BULLETIN Covering the global threat landscape

3ROS EXPLOIT FRAMEWORK KIT – ONE MORE FOR THE INFECTION ROAD!

Aditya K. Sood
Cloud Threat Labs, Elastica, USA

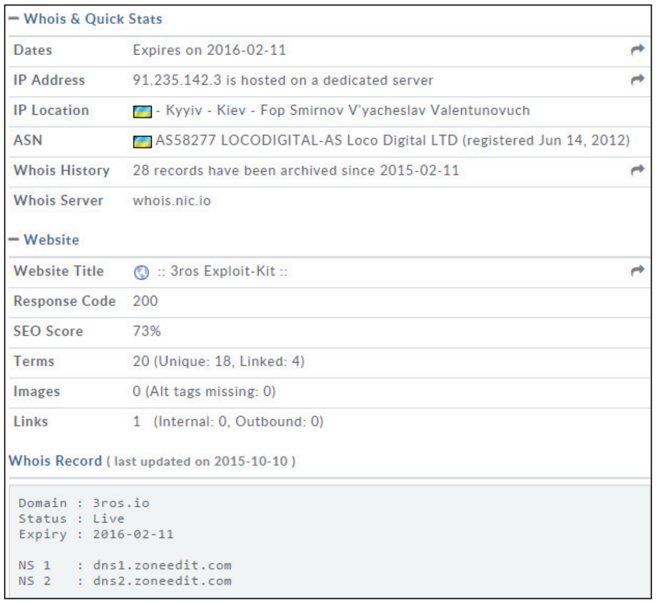
Rohit Bansal
SecNiche Security Labs, USA



INTRODUCTION

Browser exploit kits are used on a large scale to infect systems in an automated manner. 3ROS is a new entry in the world of exploit kits: another piece of crimeware designed to ease the process of carrying out infections on the Internet. The 3ROS exploit kit has some advanced operational functionalities including a very sophisticated design from a GUI perspective. The name ‘3ROS’ appears to have been taken from *Diablo III Reaper of Souls* [1], an action game published by *Blizzard Entertainment*. Researchers from *Proofpoint* have recently written about the Hunter exploit kit [2], which they believe is a rebranded version of the 3ROS exploit kit. In terms of layout, the two exploit kits do indeed look similar. We have been following the 3ROS exploit kit for the last four to five months, and it is quite possible that the same author has transformed 3ROS into the Hunter exploit kit. However, since the Hunter exploit kit is based on 3ROS, we prefer to discuss the design of the latter, because it is the primary source. In this article, we discuss 3ROS version 1.0.0.

We believe, based on indicators including the language and terminology encountered during the analysis of the 3ROS exploit kit, that it is of Russian or Ukrainian origin. We also discovered the dedicated domain 3ros.io, which we assume was for selling the exploit kit. The Whois record for the domain lists its IP location as Kiev (see Figure 1), although the domain no longer resolves.



– Whois & Quick Stats

Dates	Expires on 2016-02-11
IP Address	91.235.142.3 is hosted on a dedicated server
IP Location	- Kyyiv - Kiev - Fop Smirnov V'yacheslav Valentunovuch
ASN	AS58277 LOCODIGITAL-AS Loco Digital LTD (registered Jun 14, 2012)
Whois History	28 records have been archived since 2015-02-11
Whois Server	whois.nic.io

– Website

Website Title	:: 3ros Exploit-Kit ::
Response Code	200
SEO Score	73%
Terms	20 (Unique: 18, Linked: 4)
Images	0 (Alt tags missing: 0)
Links	1 (Internal: 0, Outbound: 0)

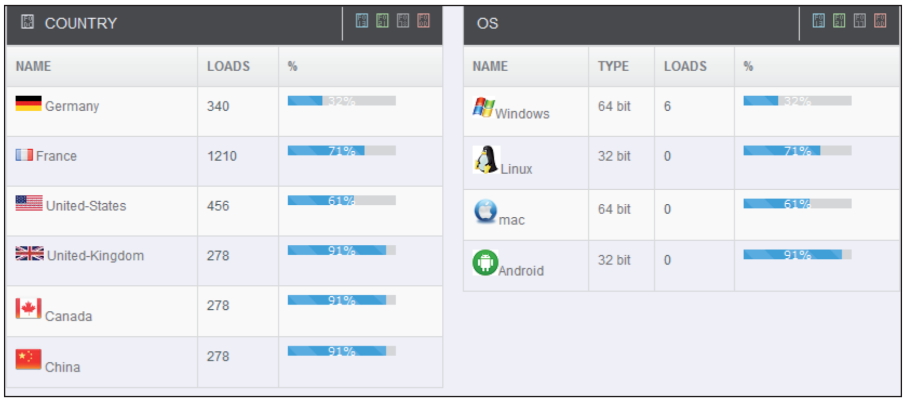
Whois Record (last updated on 2015-10-10)

```
Domain : 3ros.io
Status : Live
Expiry : 2016-02-11

NS 1   : dns1.zoneedit.com
NS 2   : dns2.zoneedit.com
```

Figure 1: Whois record for the 3ros.io domain.

In addition, when we looked at the infection statistics (shown in Figure 2), we noted that there were no infections in the Russian/Ukrainian region despite other large countries being targeted.



COUNTRY

NAME	LOADS	%
Germany	340	32%
France	1210	71%
United-States	456	61%
United-Kingdom	278	91%
Canada	278	91%
China	278	91%

OS

NAME	TYPE	LOADS	%
Windows	64 bit	6	32%
Linux	32 bit	0	71%
mac	64 bit	0	61%
Android	32 bit	0	91%

Figure 2: 3ROS infection statistics.

In the following sections, we concentrate on the 3ROS exploit kit's exploit generation and delivery process.

CUSTOM EXPLOIT GENERATION FUNCTIONALITY

3ROS provides attackers with a very smart exploit generation functionality. It has a well designed GUI that makes the steps involved in creating an exploit very straightforward. The process is as follows:

Exploit selection phase

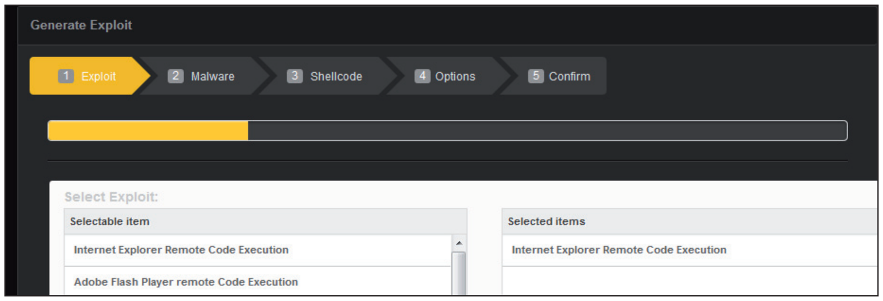
A number of exploits are provided for the 'exploit selection' phase; the attacker can pick his/her preferred exploit from a selection box. Figure 3 shows the exploit generation component, in which a number of *Internet Explorer*, *Mozilla Firefox*, *Adobe Flash*, *Oracle Java*, etc. exploits are available.

Malware selection phase

Once an exploit has been selected, the next step is to upload the malware that will be downloaded onto the targeted user system after successful exploitation. 3ROS gives full control to the attacker to build the same exploit with different malware binaries including bots, advanced spyware, ransomware or other (malicious) binaries, depending on the attacker's specific needs. Figure 4 shows the malware selection component.

Shellcode selection phase

The exploit kit also provides the option to select a variety of shellcodes, as shown in Figure 5. The shellcodes can be selected as follows:



Select Exploit:

Selectable item	Selected items
Internet Explorer Remote Code Execution	Internet Explorer Remote Code Execution
Adobe Flash Player remote Code Execution	

Figure 3: Exploit selection component.

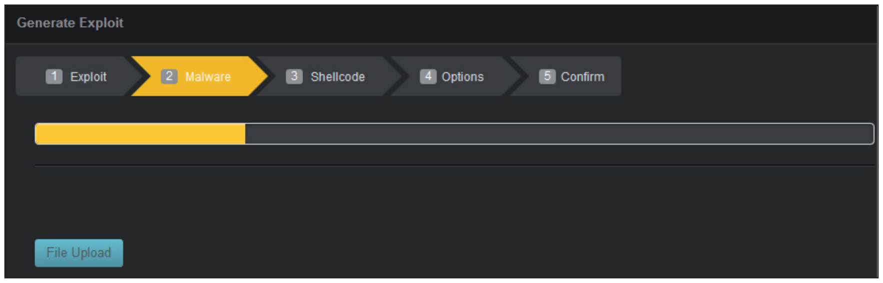


Figure 4: Malware selection component.

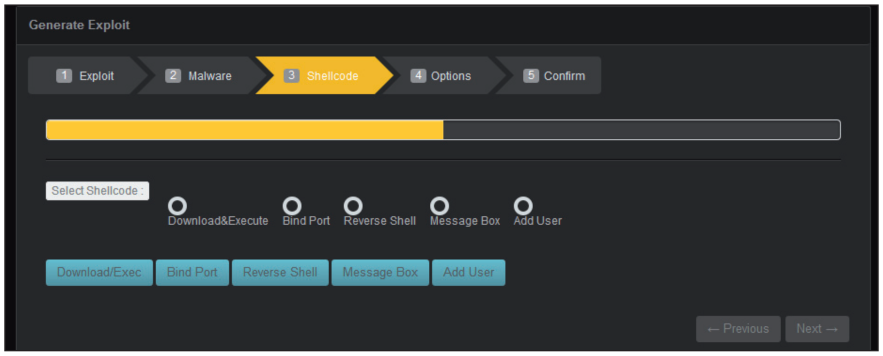


Figure 5: Shellcode selection component.

- For cases in which the attacker is more interested in gaining control of the compromised machine than in delivering malware, the exploit kit supports shellcodes for the direct binding of a shell on the provided port, executing a reverse shell, and adding an unauthorized account to the compromised machine. The attacker can easily configure the shellcode arguments for all the shellcodes by supplying information as required by specific network environments.
- If the attacker is interested in delivering and executing malware on the target system, the malware binary selected in the earlier malware selection phase is used together with 'Download&Execute' shellcode, which simply downloads the malicious binary on the target system and executes it after successful exploitation of the vulnerability in the targeted component in the browser.

Anti-detection check phase

Once the shellcode has been selected or a binary has been generated, the 3ROS exploit kit provides the functionality to check the effectiveness of the shellcode/binary against anti-virus engines and other detection tools. The anti-detection component is already embedded in this process; it is thus easy for the attacker to check whether the shellcode and malware binary can be detected. The attacker first checks the shellcode (including binary) against all anti-virus engines using the exploit kit's built-in functionality. Like the majority of exploit kits, 3ROS uses the anonymous AV checker service *Scan4You* [3] to validate and verify the detection rate of shellcode including malware binaries. If the malware is found to be detected, 3ROS provides a File Update (FUD) service, which can be used to 'update' the malware binary so that it can no longer be detected by AV engines. There is an 'Undetect' option for this purpose. Figure 6 highlights the

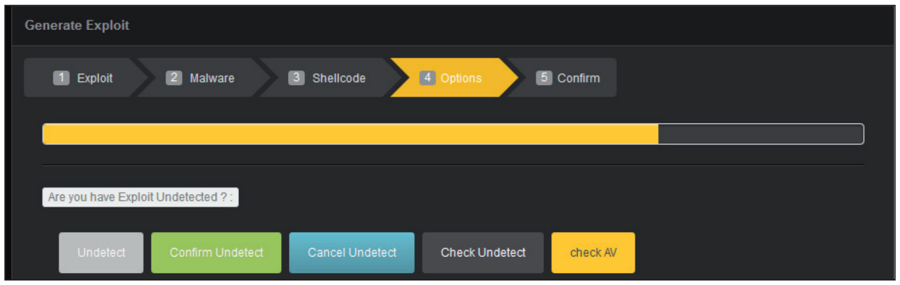


Figure 6: Malware and shellcode binary anti-detection verification.

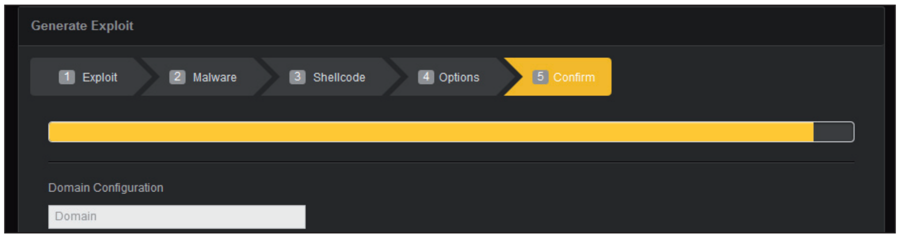
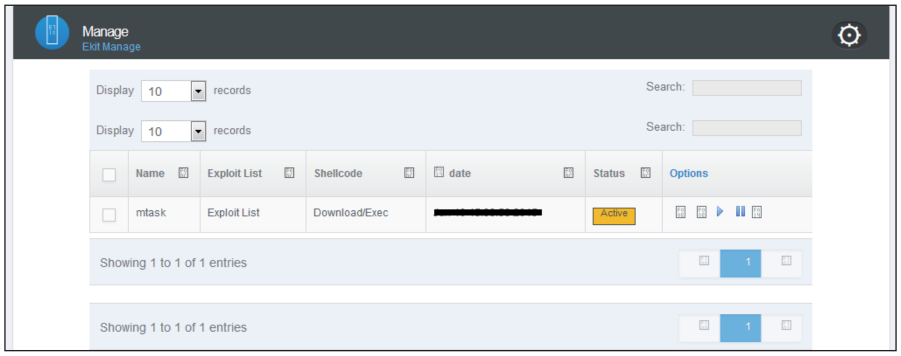


Figure 7: Domain configuration component.



	Name	Exploit List	Shellcode	date	Status	Options
☐	mtask	Exploit List	Download/Exec	[redacted]	Active	

Figure 8: Exploit task generation.

various options that are provided as part of the anti-detection check.

Domain selection phase

Once the AV check has finished, the final step, as seen in Figure 7, is to provide the domain information (primarily the domain name) relating to where the exploit will be hosted. Once this step has been completed, a new task appears on the dashboard, as shown in Figure 8, showing that the exploit is ready to be served.

The 3ROS exploit kit has a well designed notification system to tell its customers about updates such as the inclusion of new exploits, the addition of FUD services, etc.

EXPLOIT DELIVERY MECHANISM

In this section, we discuss, step by step, how the 3ROS exploit kit delivers an exploit to the target system. The exploit-serving URL is distributed to the targeted user via a spear-phishing email with a message that is crafted by the attacker in a sophisticated manner using social engineering tricks. Let's look in detail at what happens when the target clicks the embedded link pertaining to the 3ROS exploit kit.

Step 1: Once the link is clicked by the target, the browser lands on a page with the structure 'http://www.example.com/task/mtask/index.php'**.** The browser tries to fetch the resource from 3ROS's main handler website to perform the next step. 3ROS sends the HTTP response shown in Figure 9 back to the browser.

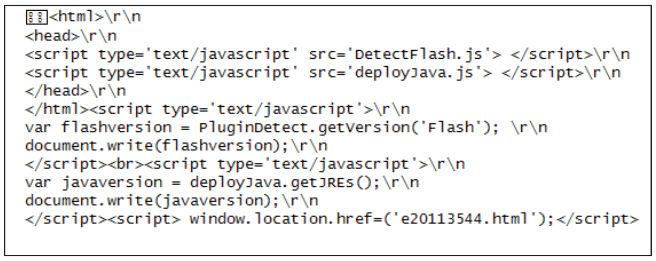
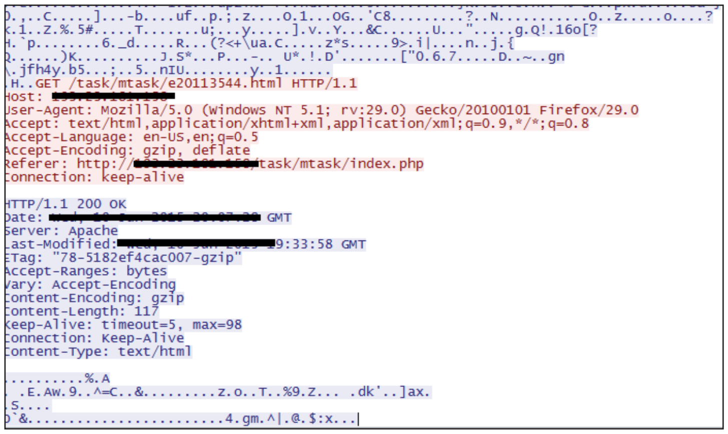

```
<html>\r\n
<head>\r\n
<script type='text/javascript' src='DetectFlash.js'> </script>\r\n
<script type='text/javascript' src='deployJava.js'> </script>\r\n
</head>\r\n
</html><script type='text/javascript'>\r\n
var flashversion = PluginDetect.getVersion('Flash'); \r\n
document.write(flashversion);\r\n
</script><br><script type='text/javascript'>\r\n
var javaversion = deployJava.getJREs();\r\n
document.write(javaversion);\r\n
</script><script> window.location.href=('e20113544.html');</script>
```

Figure 9: Pre-exploit delivery verification of plug-ins.

Step 2: The response received from the web server performs fingerprinting of the browser plug-ins, extracting the version number in order to determine whether the installed version is vulnerable against a specific vulnerability whose exploit is embedded in the exploit kit. Figure 10 shows that the Flash component ('DetectFlash.js') information is extracted through the browser using PluginDetect [4], which is a standard JavaScript library used to retrieve information about plug-ins installed on a target system. The Java component ('deployJava.js') is used to extract information about the installed version of JRE. 'deployJava.js' [5] is a JavaScript file that is provided with the Java deployment toolkit to enable the browser components to retrieve information about installed JRE so that applets and Java programs can be executed as required. At present, all browser exploit kits use PluginDetect as a primary tool for gathering information which is further used to make robust decisions in the selection and serving of exploits on the fly. The JavaScript libraries are fetched from URLs with the following format:

- http://www.example.com/task/mtask/DetectFlash.js
- http://www.example.com/task/mtask/deployJava.js

Step 3: Once a vulnerable version is detected, the browser sends a request to fetch the exploit page ('e20113544.html'), as shown in Figure 10. The request is accepted by the web server and the response is served to the browser.


```
D.,..C.....]...-b....uf..p.;.z....O.1...OG..'C8.........?..N..........O..Z.....o....?
k.1..Z.%.5#.....T.......u;...y.....].v..Y...&C......U..."......g.Q!.16o[?
H.`p........6._d.....R...(?<+\ua.C.....z*s.....9>.i|....n..j.{
Q.......)K..........J.S*...P...-.. U*.!.D'.......["0.6.7.....D..~..gn
\.jfh4y.b5...;..5..nIU........y..1......
.H..GET /task/mtask/e20113544.html HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:29.0) Gecko/20100101 Firefox/29.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://[redacted]task/mtask/index.php
Connection: keep-alive

HTTP/1.1 200 OK
Date: [redacted] GMT
Server: Apache
Last-Modified:[redacted]9:33:58 GMT
ETag: "78-5182ef4cac007-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 117
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: text/html

..........%.A
. .E.Aw.9..^=C..&.........z.o..T..%9.Z... .dk'..]ax.
.S....
D`&.......................4.gm.^|.@.$:x...|
```

Figure 10: Exploit delivery after verification of plug-ins.

Figure 11 shows that the 3ROS exploit kit served 'Song.jar' files, which were actually Java files serving exploits in compressed format. Since the test environment was running an obsolete version of JRE, the exploit was successfully served and the targeted user system was compromised.

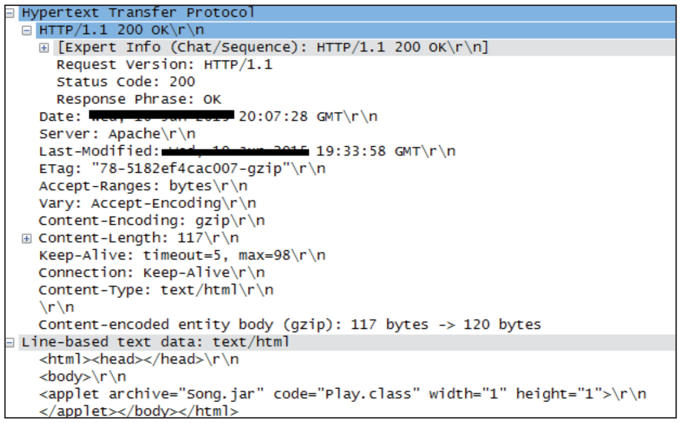

```
⊟ Hypertext Transfer Protocol
  ⊟ HTTP/1.1 200 OK\r\n
    ⊞ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Request Version: HTTP/1.1
      Status Code: 200
      Response Phrase: OK
    Date: [redacted] 20:07:28 GMT\r\n
    Server: Apache\r\n
    Last-Modified:[redacted] 19:33:58 GMT\r\n
    ETag: "78-5182ef4cac007-gzip"\r\n
    Accept-Ranges: bytes\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
  ⊞ Content-Length: 117\r\n
    Keep-Alive: timeout=5, max=98\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html\r\n
    \r\n
    Content-encoded entity body (gzip): 117 bytes -> 120 bytes
⊟ Line-based text data: text/html
    <html><head></head>\r\n
    <body>\r\n
    <applet archive="Song.jar" code="Play.class" width="1" height="1">\r\n
    </applet></body></html>
```

Figure 11: Jar exploit file.

Flash Player exploit that works on Flash 13.x, 14.x, 15.x, 16.x, 17.x and 18.x running IE 6–11 on Windows 7 and 8.1 and on XP. Also works on Firefox and Chrome.
Internet Explorer exploit that works on IE 6–11 on Windows XP, Vista, 7 and 8.1.
Java exploit that works on Java 7 update 25 and under.
Client-side exploit that includes Word and PowerPoint that works on 2003, 2007, 2010, 2013.
Silverlight exploits.

Table 1: General information about exploits served by 3ROS.

```
Alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (
msg:"Win32.Exploit.3ROS - Exploit Detection";
flow:established,from_client;
content:"GET";
http_method;content:"/task/mtask/";
http_uri; fast_pattern;
content:"Accept-Encoding: gzip, deflate";
http_header; pcre:"/\/task\/mtask\/(?:DetectFlash.js|deployJava.js|e20113544.
html)$/U"; pcre:"/(?:Firefox|MSIE)/H";
 reference:url,<Add Reference URL>;
classtype:Exploit; sid:XXXXXXXXX; rev:1; )
```

Listing 1 : Signature to detect specific 3ROS exploits.

Table 1 shows some of the exploits that are served by 3ROS.

The signature shown in Listing 1 can be used to catch specific exploits from the 3ROS exploit kit.

CONCLUSION

In this article, we have discussed primarily the design of the 3ROS exploit kit, covering specifically its exploit generation and delivery mechanisms, thereby highlighting different techniques deployed by the 3ROS authors to make the infection process smoother during drive-by download attacks. From a management perspective, the design of 3ROS is very advanced and one can expect that the same source code will be used in different ways to generate more variants.

REFERENCES

[1] https://en.wikipedia.org/wiki/Diablo_III:_Reaper_of_Souls.

[2] https://www.proofpoint.com/us/threat-insight/post/Hunter-Exploit-Kit-Targets-Brazilian-Banking-Customers.

[3] http://scan4you.net/.

[4] http://www.pinlady.net/PluginDetect/.

[5] https://www.java.com/js/deployJava.txt.

Editor: Martijn Grooten

Chief of Operations: John Hawes

Security Test Engineers: Scott James, Tony Oliveira, Adrian Luca, Ionuţ Răileanu

Sales Executive: Allison Sketchley

Editorial Assistant: Helen Martin

Developer: Lian Sebe

Consultant Technical Editor: Dr Morton Swimmer

© 2015 Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England

Tel: +44 (0)1235 555139 Fax: +44 (0)1865 543153

Email: editorial@virusbtn.com Web: https://www.virusbtn.com/