

目 录

第 1 章 Web 安全的关键点	1	2.6 一个伪装出来的世界——CSS	51
1.1 数据与指令	1	2.6.1 CSS 容错性	51
1.2 浏览器的同源策略	4	2.6.2 样式伪装	52
1.3 信任与信任关系	7	2.6.3 CSS 伪类	52
1.4 社会工程学的作用	9	2.6.4 CSS3 的属性选择符	53
1.5 攻防不单一	9	2.7 另一个幽灵——ActionScript	55
1.6 场景很重要	10	2.7.1 Flash 安全沙箱	55
1.7 小结	11	2.7.2 HTML 嵌入 Flash 的 安全相关配置	59
第 2 章 前端基础	12	2.7.3 跨站 Flash	61
2.1 W3C 的世界法则	12	2.7.4 参数传递	64
2.2 URL	14	2.7.5 Flash 里的内嵌 HTML	65
2.3 HTTP 协议	15	2.7.6 与 JavaScript 通信	67
2.4 松散的 HTML 世界	19	2.7.7 网络通信	71
2.4.1 DOM 树	20	2.7.8 其他安全问题	71
2.4.2 iframe 内嵌出一个 开放的世界	21	第 3 章 前端黑客之 XSS	72
2.4.3 HTML 内嵌脚本执行	22	3.1 XSS 概述	73
2.5 跨站之魂——JavaScript	23	3.1.1 “跨站脚本”重要的是脚本	73
2.5.1 DOM 树操作	23	3.1.2 一个小例子	74
2.5.2 AJAX 风险	25	3.2 XSS 类型	76
2.5.3 模拟用户发起浏览器请求	30	3.2.1 反射型 XSS	76
2.5.4 Cookie 安全	33	3.2.2 存储型 XSS	77
2.5.5 本地存储风险	43	3.2.3 DOM XSS	78
2.5.6 E4X 带来的混乱世界	48	3.3 哪里可以出现 XSS 攻击	80
2.5.7 JavaScript 函数劫持	49	3.4 有何危害	81

第 4 章 前端黑客之 CSRF	83	6.1.3 请求中的玄机	134
4.1 CSRF 概述	84	6.1.4 关于存储型 XSS 挖掘	135
4.1.1 跨站点的请求	84	6.2 神奇的 DOM 渲染	135
4.1.2 请求是伪造的	84	6.2.1 HTML 与 JavaScript	
4.1.3 一个场景	84	自解码机制	136
4.2 CSRF 类型	89	6.2.2 具备 HtmlEncode	
4.2.1 HTML CSRF 攻击	89	功能的标签	140
4.2.2 JSON HiJacking 攻击	90	6.2.3 URL 编码差异	142
4.2.3 Flash CSRF 攻击	94	6.2.4 DOM 修正式渲染	145
4.3 有何危害	96	6.2.5 一种 DOM fuzzing 技巧	146
第 5 章 前端黑客之界面操作劫持	97	6.3 DOM XSS 挖掘	150
5.1 界面操作劫持概述	97	6.3.1 静态方法	150
5.1.1 点击劫持 (Clickjacking)	98	6.3.2 动态方法	151
5.1.2 拖放劫持		6.4 Flash XSS 挖掘	153
(Drag&Dropjacking)	98	6.4.1 XSF 挖掘思路	153
5.1.3 触屏劫持 (Tapjacking)	99	6.4.2 Google Flash XSS 挖掘	156
5.2 界面操作劫持技术原理分析	99	6.5 字符集缺陷导致的 XSS	159
5.2.1 透明层+iframe	99	6.5.1 宽字节编码带来的安全问题	160
5.2.2 点击劫持技术实现	100	6.5.2 UTF-7 问题	161
5.2.3 拖放劫持技术实现	101	6.5.3 浏览器处理字符集编码	
5.2.4 触屏劫持技术实现	103	BUG 带来的安全问题	165
5.3 界面操作劫持实例	106	6.6 绕过浏览器 XSS Filter	165
5.3.1 点击劫持实例	106	6.6.1 响应头 CRLF 注入绕过	165
5.3.2 拖放劫持实例	111	6.6.2 针对同域的白名单	166
5.3.3 触屏劫持实例	119	6.6.3 场景依赖性高的绕过	167
5.4 有何危害	121	6.7 混淆的代码	169
第 6 章 漏洞挖掘	123	6.7.1 浏览器的进制常识	169
6.1 普通 XSS 漏洞自动化		6.7.2 浏览器的编码常识	175
挖掘思路	124	6.7.3 HTML 中的代码注入技巧	177
6.1.1 URL 上的玄机	125	6.7.4 CSS 中的代码注入技巧	190
6.1.2 HTML 中的玄机	127	6.7.5 JavaScript 中的代码	
		注入技巧	196

6.7.6	突破 URL 过滤	201	7.7.1	浏览器<script>请求	247
6.7.7	更多经典的混淆 CheckList	202	7.7.2	浏览器跨域 AJAX 请求	248
6.8	其他案例分享—— Gmail Cookie XSS	204	7.7.3	服务端 WebSocket 推送指令	249
第 7 章	漏洞利用	206	7.7.4	postMessage 方式推送指令	251
7.1	渗透前的准备	206	7.8	真实案例剖析	254
7.2	偷取隐私数据	208	7.8.1	高级钓鱼攻击之百度空间 登录 DIV 层钓鱼	254
7.2.1	XSS 探针: xssprobe	208	7.8.2	高级钓鱼攻击之 Gmail 正常服务钓鱼	261
7.2.2	Referer 惹的祸	214	7.8.3	人人网跨子域盗取 MSN 号	265
7.2.3	浏览器记住的明文密码	216	7.8.4	跨站获取更高权限	267
7.2.4	键盘记录器	219	7.8.5	大规模 XSS 攻击思想	275
7.2.5	偷取黑客隐私的 一个小技巧	222	7.9	关于 XSS 利用框架	276
7.3	内网渗透技术	223	第 8 章	HTML5 安全	277
7.3.1	获取内网 IP	223	8.1	新标签和新属性绕过 黑名单策略	278
7.3.2	获取内网 IP 端口	224	8.1.1	跨站中的黑名单策略	278
7.3.3	获取内网主机存活状态	225	8.1.2	新元素突破黑名单策略	280
7.3.4	开启路由器的远程 访问能力	226	8.2	History API 中的新方法	282
7.3.5	内网脆弱的 Web 应用控制	227	8.2.1	pushState()和 replaceState()	282
7.4	基于 CSRF 的攻击技术	228	8.2.2	短地址+History 新方法= 完美隐藏 URL 恶意代码	283
7.4.1	基于 CSRF 的 XSS 攻击	229	8.2.3	伪造历史记录	284
7.5	浏览器劫持技术	230	8.3	HTML5 下的僵尸网络	285
7.6	一些跨域操作技术	232	8.3.1	Web Worker 的使用	286
7.6.1	IE res:协议跨域	232	8.3.2	CORS 向任意网站 发送跨域请求	287
7.6.2	CSS String Injection 跨域	233	8.3.3	一个 HTML5 僵尸网络实例	287
7.6.3	浏览器特权区域风险	235	8.4	地理定位暴露你的位置	290
7.6.4	浏览器扩展风险	237	8.4.1	隐私保护机制	290
7.6.5	跨子域: document.domain 技巧	240	8.4.2	通过 XSS 盗取地理位置	292
7.6.6	更多经典的跨域索引	245			
7.7	XSS Proxy 技术	246			

第 9 章 Web 蠕虫	293	9.4.4 GoogleReader 的 ShareJacking 蠕虫	327
9.1 Web 蠕虫思想	294	9.4.5 ClickJacking 蠕虫 爆发的可能性	335
9.2 XSS 蠕虫	295	第 10 章 关于防御	336
9.2.1 原理+一个故事	295	10.1 浏览器厂商的防御	336
9.2.2 危害性	297	10.1.1 HTTP 响应的 X-头部	337
9.2.3 SNS 社区 XSS 蠕虫	300	10.1.2 迟到的 CSP 策略	338
9.2.4 简约且原生态的蠕虫	304	10.2 Web 厂商的防御	341
9.2.5 蠕虫需要追求原生态	305	10.2.1 域分离	341
9.3 CSRF 蠕虫	307	10.2.2 安全传输	342
9.3.1 关于原理和危害性	307	10.2.3 安全的 Cookie	343
9.3.2 译言 CSRF 蠕虫	308	10.2.4 优秀的验证码	343
9.3.3 饭否 CSRF 蠕虫—— 邪恶的 Flash 游戏	314	10.2.5 谨慎第三方内容	344
9.3.4 CSRF 蠕虫存在的 可能性分析	320	10.2.6 XSS 防御方案	345
9.4 ClickJacking 蠕虫	324	10.2.7 CSRF 防御方案	348
9.4.1 ClickJacking 蠕虫的由来	325	10.2.8 界面操作劫持防御	353
9.4.2 ClickJacking 蠕虫 技术原理分析	325	10.3 用户的防御	357
9.4.3 Facebook 的 LikeJacking 蠕虫	327	10.4 邪恶的 SNS 社区	359