নিরাপদ কোডিং অভ্যাস

জাবেদ মোর্শেদ চৌধুরী

নিরাপদ কোডিং হচ্ছে নিরাপদ সফটওয়্যার তৈরির পূর্বশর্ত। বেশিরভাগ প্রোগ্রামারই কোনো সফটওয়্যারের ফাংশনালিটি নিয়ে বেশি ভাবেন। সাধারণত নিরাপত্তার বিষয়টি সফটওয়্যার ডেভেলপমেন্ট সাইকেলের শেষের দিকে অ্যাড-হক হিসেবে চিন্তা করা হয়, যা একটি ভুল প্রক্রিয়া।নিরাপদ সফটওয়্যার তৈরির জন্য প্রথম থেকেই এবং প্রতিটি স্তরেই (যেমন : রিকোয়ারমেন্ট, ডিজাইন, ইমপ্লিমেন্টেশন/কোডিং, টেস্টিং ও ডিপ্লোয়মেন্ট) নিরাপত্তার বিষয়টি মাথায় রাখতে হবে। একজন প্রোগ্রামারের নিরাপত্তা বিষয়ক অজ্ঞতার ও অনিরাপদ কোডিংয়ের জন্য সফটওয়্যারটি সহজেই কম্প্রোমাইজ বা হ্যাক হতে পারে। এক গবেষণায় দেখা গেছে, নিরাপদ কোডিং প্র্যাকটিস ৫০ শতাংশ হ্যাকের ঘটনা কমিয়ে ফেলে। অন্য একটি গবেষণায় দেখা গেছে, পাঁচ আকারে কোনো সফটওয়্যারের সিকিউরিটি হোল বন্ধ করতে ৬০শতাংশ বেশি খরচ হয়।

এই লেখায় নিরাপদ কোডিং নিয়ে আলোচনা করা হয়েছে। তবে এর আগে অনিরাপদ কোডিংয়ের জন্য কী কী সমস্যা হতে পারে, তা দেখব।

বাফার ওভার ফ্লো : বাফার ওভার ফ্লো হয় তখন, যখন কোনো একটি প্রোগ্রামের কোনো ইনপুট তার বরাদ্দ মেমরির চেয়ে বেশি জায়গায় লিখতে পারে। কোনো একজন হ্যাকার বাফার ওভার ফ্লো ব্যবহার করে পুরো প্রোগ্রামের কন্ট্রোল নিয়ে নিতে পারে বা প্রোগ্রামটি ক্র্যাশ করিয়ে দিতে পারে। সি ও সি++ ল্যাঙ্গুয়েজ সাধারণত বাফার ওভার ফ্লোতে বেশি আক্রান্ত হয়। জাভাতে অ্যারে বাউন্ড ফাংশনালিটির কারণে সরাসরি মেমরি অ্যাক্সেস করা যায় না। তাই জাভা সাধারণত বাফার ওভার ফ্লোতে কম আক্রান্ত হয়।

ইন্টিজার ওভার ফ্লো : ইন্টিজার ওভার ফ্লো হয় তখন, যখন কোনো ইন্টিজার ভেরিয়েবল নিজের স্টোরেজ ক্ষমতার চেয়ে বড় সংখ্যাকে স্টোর করাতে চায়। এটা সাধারণত দু'টি সংখ্যার যোগফল বা গুণফল হিসেবে হতে পারে (যেমন : a= a+ b)। সি ও সি++ ল্যাঙ্গুয়েজে সাধারণত ইন্টিজার ওভার ফ্লো বেশি আক্রান্ত হয়। জাভাতে রেঞ্জ চেক বলে একটি ফাংশনালিটির মাধ্যমে এর মাত্রা কমিয়ে আনা হয়েছে।

```
/** short init number =0;
char buffer[large_value];
while (number < MAX_NUM)
  {
  number += getInput (buffer+number);
  }
**/
```

এই উদাহরণে 'number' ভেরিয়েবলটি MAX_NUM-এর চেয়ে ছোট হলে তা একসময় ইন্টিজার ওভার ফ্লো সমস্যার তৈরি করবে। এর ফলে MAX_Num-1তম বাইট ওভার রাইট হয়ে যাবে।

ফরমেট স্ট্রিং অ্যাটাক : এ ধরনের সমস্যাযুক্ত সফটওয়্যারে হ্যাকারেরা প্রোগ্রামে ত্রুটিপূর্ণ ইনপুট দিয়ে থাকে। ইনপুটটি একটি কমান্ড হিসেবে কমপিউটার সিস্টেমে কাজ করে। এর মাধ্যমে হ্যাকার তথ্য চুরি, অন্য কোনো কোড রান করা ও কমপিউটারের কন্ট্রোল নিয়ে নিতে পারে।

```
/**
int main ( int argv, char  * argv[])
  {
   printf (agrv[1]);
   return 0;
  }
**/
```

উপরোল্লিখিত প্রোগ্রামে যদি কেউ %x বা %n ধরনের ইনপুট দেয়, তবে প্রোগ্রামটি অপ্রত্যাশিত ফল প্রদর্শন করবে। printf(argv[1])-এর স্থলে printf("%s", argv[1]) ব্যবহার করলে ভালনারিবিলিটি কিছুটা কমবে।

ক্রস সাইট স্ক্রিপটিং : ক্রস সাইট স্ক্রিপটিং সমস্যাগুলো সাধারণত ওয়েবসাইটগুলোতে পাওয়া যায়। এর মাধ্যমে হ্যাকারেরা সাধারণত ম্যালেশিয়াস ডাটা পাঠায়। ফলে এরা অ্যাক্সেস কন্ট্রোল সিস্টেমকে বাইপাস করতে সক্ষম হয়। এতে করে ভিকটিমের ওয়েব ব্রাউজারে ম্যালেশিয়াস ডাটা দেখা যায়। অনেক সময় স্থায়ীভাবেও ম্যালেশিয়াস ডাটা কোনো ওয়েবসাইটে স্টোর করা যায়। এ ধরনের আক্রমণের মাধ্যমে হ্যাকারেরা ওয়েবসাইট ডিফেস, কুকি চুরি, গুরুত্বপূর্ণ তথ্য চুরি বা ফিশিং অ্যাটাক করে থাকে।

এসকিউএল ইনজেকশন : এসকিউএল ইনজেকশন হলো এসকিউএল কমান্ড/কোয়ারি, যা ব্যবহারকারীর দেয়া তথ্যের সাথে এমনভাবে যুক্ত হয়, যাতে করে এটি অ্যাক্সেস কন্ট্রোল সিস্টেমকে বাইপাস করে ডাটাবেজের তথ্য সংযোজন, বিয়োজন বা প্রদর্শন করতে পারে। এই আক্রমণের মাধ্যমে হ্যাকার ডাটাবেজে থাকা যেকোনো তথ্য চুরি করতে পারে। এটা হতে পারে ব্যক্তিগত তথ্য, ক্রেডিট কার্ড নম্বর অথবা অন্য কোনো সংবেদনশীল তথ্য।

```
<?php 5
<form method='post' action ='Login_Account.php'>
<input type = 'text' name= 'username'>
<input type = 'password' name= 'password'>
</form>
?>
```

উপরোল্লিখিত এইচটিএমএল স্ক্রিপ্টে একটি বেসিক অথেনটিকেশন মেথড দেখানো হয়েছে। ব্যবহারকারীর ক্রাডেশিয়াল (Cradential) (ইউজার নেম ও পাসওয়ার্ড) Login_Account.php ফাইলের মাধ্যমে পাঠানো হয়। ঠিকমতো ইনপুট ভ্যালিডেশন করা না হলে এ ধরনের ভালনারিবিলিটিকে ব্যবহার করে ম্যালেশিয়াস এসকিউএল স্টেটমেন্টের (যেমন : Select * from LOGIN where username='john_smith' and password = ' ' or 1=1;) মাধ্যমে অথেনটিকেশন মেথডকে বাইপাস করা সম্ভব।

অনিরাপদভাবে সরাসরি অবজেক্ট রেফারেন্সিং : অনেক সময় প্রোগ্রামারেরা সঠিক অথরাইজেশন ব্যবহার না করে কোনো একটি রিসোর্সের যেমন : ইউআরএল, ফরমেট প্যারামিটার বা ডাটাবেজ রেকর্ডকে প্রোগ্রামের ভেতরের অন্য কোনো মডিউলে ব্যবহার করেন। এতে একজন হামলাকারী যার ওই রিসোর্সের ওপর অথরাইজেশন নেই, সেও এই রিসোর্সটি ব্যবহার করতে পারে এবং ম্যানিপুলেট করতে পারে।

```
/** Example of Insecure Direct Object Reference **/
http://www.abc.com/resources/accounts/information/getinfo.jsp?padeId=help.html
```

এ ধরনের ইউআরএল দিয়ে রিসোর্স অ্যাক্সেস করার ক্ষেত্রে যদি সঠিক অথরাইজেশন ব্যবহার করা না হয়, তবে হ্যাকারেরা ডিরেক্টরি ব্রাউজিংয়ের মাধ্যমে অন্য ফোল্ডারের ডাটা অ্যাক্সেস করতে পারে, যা তাদের অ্যাক্সেস করতে পারার কথা নয়।

সঠিকভাবে এরর হ্যান্ডলিং না করা : যদি সঠিকভাবে এরর হ্যান্ডলিং করা না হয়, তবে অনেক সময় অনেক গুরুত্বপূর্ণ তথ্য প্রকাশ পেয়ে যেতে পারে। এই ধরনের তথ্য হ্যাকারেরা ব্যবহার করে থাকেন তাদের আক্রমণ প্রক্রিয়া ঠিক করার সময়। ভুলভাবে এরর হ্যান্ডলিংয়ের ফলে সিস্টেম ক্র্যাশ, টার্মিনেট অথবা রিস্টার্ট হয়ে যেতে পারে। প্রায় প্রত্যেক জনপ্রিয় প্রোগ্রামিং ল্যাঙ্গুয়েজের এক্সসেপশন হ্যান্ডলিংয়ের ম্যাকানিজম আছে, যা দিয়ে ▸

অনাকাঙ্ক্ষিত ডাটা বের হয়ে যাওয়া থেকে প্রোগ্রামকে রক্ষা করা সম্ভব।

```
/** Example of Improper Error handling and information Leakage **/
404 Not Found
Not Found
The requested URL /abc/xyz_help/ was not found on this server
Apache/ 2.2.3(Debian) PHP/5.2.0-8+etch13 mod_ssl/2.2.3 OpenSSL/0.9.8c server at abc.pqr.de port 80
```

এই উদাহরণে এরর মেসেজটি ওয়েব সার্ভার, অপারেটিং সিস্টেম, পোর্ট নম্বর, পিএইচপি ভার্সনসহ অন্যান্য তথ্য প্রকাশ করে দিচ্ছে।

নিরাপদ কোডিংয়ের জন্য কিছু নির্দেশনা : ০১. ডিজাইন ফেজেই সিকিউরিটি দুর্বলতা ও কৌশলগুলো শনাক্ত করা। মনে রাখা দরকার, ডিজাইন ফেজে কোনো দুর্বলতা ধরা পড়লে তা ফিক্স করা অনেক সহজ ও সাশ্রয়ী। ০২. সঠিক ও কার্যকরভাবে থ্রেট মডেলিং করা। অর্থাৎ সিস্টেমের জন্য সম্ভাব্য সমস্যাগুলো মডেল করা এবং যেসব ভালনারিবিলিটির জন্য সমস্যাগুলো হতে পারে, তা শনাক্ত করা। ০৩. সমসময় সঠিকভাবে ইনপুট ভ্যালিডেশন করা। ওয়েব অ্যাপ্লিকেশনের জন্য ফ্রন্ট ও ব্যাকএন্ড দুই জায়গায়ই ইনপুট ভ্যালিডেশন করা উচিত। ০৪. কার্যকর এরর হ্যান্ডলিং ম্যাকানিজম ব্যবহার করা যাতে কোনোভাবেই সিস্টেমের কোনো গোপন তথ্য বের হয়ে না যায় বা সিস্টেম অস্বাভাবিক আচরণ না করে। ০৫. কোনো প্রসেস বা মডিউলকে অ্যাক্সেস দেয়ার সময় সতর্ক থাকতে হবে যাতে সঠিক বৈধ মাধ্যম অনুসরণ করা হয়। ০৬. অবশ্যই নিরাপদ কোডিং অভ্যাস গড়ে তুলতে হবে ও সবসময় তা অনুসরণ করতে হবে।

সিকিউরিটি টেস্টিং

কোড রিভিও : পেয়ার কোডিং রিভিউ বা কোয়ালিটি টেস্টিং সফটওয়্যার ডেভেলপমেন্ট সাইকেলে গুরুত্বপূর্ণ অংশ। কিন্তু প্রথাগত রিভিউ সবসময় সিকিউরিটি হোল বা ফাংশনালিটি সঠিকভাবে পরখ করতে পারে না। তাই রিভিউয়ারকে বা রিভিউয়ার টিমের সদস্যদের সিকিউরিটি সম্পর্কে জ্ঞান থাকা আবশ্যক। রিভিউয়ের জন্য সিকিউরিটি কোড রিভিউ টুল ব্যবহার করা যেতে পারে। যেমন : PREfast বা Flawfinder।

পেন টেস্টিং : পেন টেস্টার কোনো সিস্টেম বা অ্যাপ্লিকেশনের ব্লাক বক্স টেস্টিংয়ের মাধ্যমে সিকিউরিটি পরীক্ষা করে থাকেন। একজন পেন টেস্টারের সোর্স কোড সম্পর্কে বা সিস্টেমের আর্কিটেকচার সম্পর্কে কোনো ধারণা থাকে না।

একজন পেন টেস্টার থার্ড পার্টি হিসেবে সিস্টেমের ভালনারিবিলিটি বের করার চেষ্টা করেন। এ কাজে তিনি বিভিন্ন টুল ব্যবহার করে থাকেন, যেমন : নেক্সাস, মেটাইসপ্লয়েট বা বার্প স্যুইট।

ফাজ টেস্টিং : ফাজ টেস্টিংয়ের মাধ্যমে সিস্টেমে ভুল ইনপুট দিয়ে পরীক্ষা করা হয় ও দেখা হয় সিস্টেমটি কেমন আচরণ করছে। ফাজ টেস্টিং ও অনেকটা ব্লাক বক্স টেস্টিংয়ের মতোই। প্রত্যেকবার বাগ রিপোর্ট হলে বা সিস্টেম আপডেট হলে ফাজ টেস্টিং করা উচিত। এর মাধ্যমে কমন ভালনারিবিলিটি যেমন : বাফার ওভার ফ্লো, ক্রস সাইট স্ক্রিপটিং, এসকিউএল ইনজেকশন শনাক্ত করা যায়।

শেষের কথা

সিকিউরিটি একটি ইন্টিগ্রেটেড প্রসেস। এর বিভিন্ন উপাদান রয়েছে। যেমন : সার্ভার সিকিউরিটি, অ্যাপ্লিকেশন সিকিউরিটি ইত্যাদি। আমরা সবসময় কোনো সিস্টেমের সিকিউরিটির কথা চিন্তা করলে অ্যাক্সেস কন্ট্রোল সিস্টেম, ফায়ারওয়াল এসব চিন্তা করি। বেশিরভাগ সময় আমাদের সিস্টেমটি নিরাপদভাবে কোড করা হয়েছে কি না তা পরীক্ষা করি না। ফলে অনেক সময় আমাদের প্রোগ্রামের কোডে থাকা কোনো নিরাপত্তা ছিদ্র কাজে লাগিয়ে হ্যাকাররা আমাদের পুরো সিস্টেমটিকে ধ্বংস বা নষ্ট করে ফেলতে পারে। নিরাপত্তা বিষয়ক অধ্যয়নে আমাদের মাঝে একটা প্রচলিত কথা হলো : 'It's only as strong as its weakest link.'।

ফিডব্যাক : jabedmorshed@yahoo.com