Risk-Assessment							CVSSv3 Base Score								Stellungnahme			Risikobehandlung			
Nr.	Gruppe	Feststellung	Erklärung	Empfehlung	Betroffen	Risiko	Attack Vector (AV)	Attack Complexity (AC)	Privileges Required (PR)	User Interaction (UI)	Scope (S)	Confidentiality (C)	Integrity (I)	Availability (A)	Kunde	SVA	Dienstleister	Behandlung	Maßnahme	Zieldatum	Zuständig
101	Web-Anwendung	Arbitrary File Upload / Remote Code Execution	Bei der Analyse der GUI-Implementierung in der Homematic CCU2 wurde eine Directory Traversal Schwachstelle identifiziert. Ein nicht authentifizierter Angreifer kann die Schwachstelle ausnutzen, um beliebige Dateien im Dateisystem der Homematic CCU2 zu überschreiben. Das System bietet eine JSON-API mit verschiedenen Methoden an. Einige der Methoden verlangen eine Authentifizierung für die Benutzer und andere nicht. Ein Code-Review der Datei `api/methods/user/setlanguage.tcl` zeigte, dass die Methode `User.setLanguage` einen nicht authentifizierten Angreifer erlaubt, mittels eines Directory Traversals beliebige Dateien auf der Homematic CCU2 zu überschreiben. ~~~tcl exec echo $args(userLang)>/etc/config/userprofiles/$args(userName).lang ~~~~ *Info:* [CVE-2018-7300](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7300) und http://atomic111.github.io/article/homematic-ccu2-filewrite	Es wird empfohlen, ein Authentifizierung für *alle* Methoden zu implementieren.	192.168.2.65	sehr hoch	Network (N)	Low (L)	None (N)	None (N)	Unchanged (U)	High (H)	High (H)	High (H)							
102	Web-Anwendung	Nicht vertrauenswürdige Addon Installation	Bei der Analyse der GUI-Implementierung der Homematic CCU2, insbesondere dem Addon-Management und der Erstellung eigener Addon-Pakete, wurde eine Remote Code Execution Schwachstelle festgestellt. Diese ist insbesondere dann problematisch, wenn Benutzer aus nicht vertrauenswürdigen Quellen Addon-Pakete installieren. *Prozess:* 1. Erstellen eines update_script mit dem entsprechenden Schadcode 2. Packen des Addon-Paketes 3. Hochladen und Installation des Addon-Paketes über die Homematic CCU2 GUI *Info:* [CVE-2018-7299](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7299) und http://atomic111.github.io/article/homematic-ccu2-untrusted_addon	Es wird empfohlen, ein Signaturschema für vertrauenswürdige Addon-Pakete einzuführen und den Benutzer zu warnen, wenn er nicht vertrauenswürdige oder unsignierte Addons installiert.	192.168.2.65	mittel	Local (L)	Low (L)	Low (L)	Required (R)	Unchanged (U)	Low (L)	Low (L)	Low (L)							
103	Web-Anwendung	Herunterladen der Firmware über Plain HTTP	Bei der Analyse der Homematic CCU2 Software wurde festgestellt, dass ein Cron Job Eintrag regelmäßig das Skript `/usr/local/etc/config/addons/mh/loopupd.sh` aufruft. Dieses Skript lädt regelmäßig Software-Pakete herunter, die nicht signiert sind oder anderweitig validiert werden. Somit kann ein Angreifer z. B. über DNS-Spoofing die Kommunikation umleiten und das entsprechende Software-Paket mit seinem Schadcode auf der Homematic installieren. ~~~bash ADDONDIR=/usr/local/etc/config/addons/mh #altes Schluessel-Paket loeschen v1=`rm $ADDONDIR/vpnkey_ccu2.tar.gz` v1b=`rm $ADDONDIR/newver` # #das Paket des Benutzers herunter laden v2=`/usr/bin/wget -O $ADDONDIR/newver 'http://www.meine-homematic.de/getverv2.php?id=29294&key=s35m89s3k73e19y7' -q` ~~~~ *Info:* [CVE-2018-7298](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7298) und http://atomic111.github.io/article/homematic-ccu2-firmware-via-plain-http	Es wird empfohlen, die Software-Pakete über eine verschlüsselte Verbindung (HTTPS) herunterzuladen oder über andere kryptographische Schutzfunktionen (z. B. Prüfsummen) zu schützen.	192.168.2.65	mittel	Network (N)	High (H)	None (N)	None (N)	Unchanged (U)	Low (L)	Low (L)	None (N)							
104	Web-Anwendung	Directory Traversal / Arbitrary File Read	Bei der Analyse der GUI-Implementierung in der Homematic CCU2 wurde eine Directory Traversal Schwachstelle identifiziert. Ein nicht authentifizierter Angreifer kann die Schwachstelle ausnutzen, um beliebige Dateien im Dateisystem der Homematic CCU2 auszulesen. Das System bietet eine JSON-API mit verschiedenen Methoden an. Einige der Methoden verlangen eine Authentifizierung der Benutzer, andere nicht. Ein Code-Review der Datei `api/methods/user/getlanguage.tcl` zeigte, dass die Methode `User.getLanguage` einen nicht authentifizierten Angreifer erlaubt, die erste Zeile einer beliebigen Datei auf der Homematic CCU2 auszulesen. ~~~tcl if {[catch {set fp [open "/etc/config/userprofiles/$args(userName).lang" r]] == 0} { set data [read $fp] set lang [split $data "\n"] close $fp } else { set lang "0" } jsonrpc_response [lindex $lang 0] ~~~~ *Info:* [CVE-2018-7296](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7296) und http://atomic111.github.io/article/homematic-ccu2-fileread	Es wird empfohlen, ein Authentifizierung für *alle* Methoden zu implementieren.	192.168.2.65	hoch	Network (N)	Low (L)	None (N)	None (N)	Unchanged (U)	High (H)	None (N)	None (N)							
105	Web-Server	Web-Sever mit root-Rechten gestartet	Der Web-Server und die darauf aufbauende Anwendung werden mit root-Rechten ausgeführt. Dies hat zur Folge, dass durch eine Schwachstelle im Web-Server oder Anwendung ein Angreifer sofort seinen Schadcode mit root-Rechten auf dem System ausführt. Somit kann ein Angreifer z. B. die `/etc/shadow` auslesen.	Es wird empfohlen, das komplette System zu härten, d. h. den Web-Server als ein nicht privilegierten Benutzer zu starten. *Info:* https://github.com/dev-sec	192.168.2.65	sehr hoch	Network (N)	Low (L)	None (N)	None (N)	Unchanged (U)	High (H)	High (H)	High (H)							
106	Zertifikate	Nicht vertrauenswürdiges Zertifikat	* Es wird ein gültiges X.509 Zertifikat für die Domain verwendet. * Das Zertifikat wurde von keiner anerkannten Certification Authority ausgestellt (selbst signiertes Zertifikat). * Der Common Name ist 192.168.2.65 * Die Schlüsselgröße ist 2048 Bit. * Das Zertifikat wurde mit einem SHA1 signiert. * Das Zertifikat ist bis 2020-06-26 gültig. * Das Zertifikat ist gültig und entspricht nicht den Sicherheitsanforderungen.	Es wird empfohlen, ein gültiges Zertifikat zu nutzen. Es sollte geprüft werden, ob ein öffentliches Zertifikat, d.h. welches durch eine anerkannte CA ausgestellt wurde, verwendet werden kann. Die Schlüsselgröße sollte min. __4096__ Bit und mit min. einem __SHA256__ signiert sein.	192.168.2.65	mittel	Network (N)	High (H)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
107	HTTP Header	Fehlender Content-Security-Policy Header "default-src"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "default-src" wird die Quelle für Inhalte, die der Browser nachladen darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "default-src" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "default-src 'self' cdn.example.com;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							

108	HTTP Header	Fehlender Content-Security-Policy Header "script-src"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "script-src" wird die Quelle für JavaScript-Inhalte, die der Browser nachladen darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "script-src" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "script-src 'self' js.example.com;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
109	HTTP Header	Fehlender Content-Security-Policy Header "style-src"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "style-src" wird die Quelle für CSS-Stylesheets, die der Browser nachladen darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "style-src" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash Header set Content-Security-Policy "style-src 'self' css.example.com;" # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "style-src 'self' css.example.com;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
110	HTTP Header	Fehlender Content-Security-Policy Header "img-src"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "img-src" wird die Quelle für Bilddateien, die der Browser nachladen darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "img-src" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "img-src 'self' img.example.com;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
111	HTTP Header	Fehlender Content-Security-Policy Header "connect-src"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "connect-src" werden die möglichen Gegenstellen für XMLHttpRequests (AJAX), WebSockets und EventSource bestimmt, die der Browser für weitere Verbindungen verwenden darf. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "connect-src" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "connect-src 'self'") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
112	HTTP Header	Fehlender Content-Security-Policy Header "font-src"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "font-src" wird die Quelle für Schriftartendateien, die der Browser nachladen darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "font-src" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "font-src 'self' font.example.com;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
113	HTTP Header	Fehlender Content-Security-Policy Header "object-src"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "object-src" wird die Quelle für Plugins, die der Browser nachladen darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "object-src" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "object-src 'self';") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
114	HTTP Header	Fehlender Content-Security-Policy Header "media-src"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "media-src" wird die Quelle für Audio- und Video-Inhalte für HTML5, die der Browser nachladen darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollten der Content-Security-Policy Header "media-src" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "media-src 'self' media.example.com;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
115	HTTP Header	Fehlender Content-Security-Policy Header "sandbox"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "sandbox" wird die Sandbox-Umgebung des Browsers konfiguriert, in dem Funktionen, wie das Senden von Formulardaten oder das Ausführen von JavaScript explizit erlaubt wird. Wird in der Web-Applikation die direktive "sandbox" verwendet, so werden diese Inhalte in einer Umgebung ausgeführt, die keine Inhalte außerhalb der aktuellen Quelle und keinen Aufbau weiterer Kommunikation aus dieser Umgebung ermöglicht. Dies wird häufig in iFrames eingesetzt und lässt sich über diesen Header definieren.	Es sollte der Content-Security-Policy Header "sandbox" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "sandbox allow-forms allow-scripts;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
116	HTTP Header	Fehlender Content-Security-Policy Header "report-uri"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "report-uri" wird eine URL definiert, zu der der Browser Verstöße gegen die Content-Security-Policy, die in der Web-Applikation auftreten, gemeldet werden.	Es sollte der Content-Security-Policy Header "report-uri" gesetzt werden. Für lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "report-uri /some-report-uri;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
117	HTTP Header	Fehlender Content-Security-Policy Header "child-src"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "child-src" wird die Quelle für Frames und iFrames, die der Browser verwenden darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "child-src" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "child-src 'self';") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							

118	HTTP Header	Fehlender Content-Security-Policy Header "form-action"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "form-action" werden erlaubte Ziele für Formulardaten, die der Browser verwenden darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "form-action" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "form-action 'self';") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
119	HTTP Header	Fehlender Content-Security-Policy Header "frame-ancestors"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administratoren, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "iframe-ancestor" werden erlaubte Quellen für eingebettete Inhalte, die der Browser verwenden darf, definiert. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollten der Content-Security-Policy Header gesetzt "frame-ancestors" werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "frame-ancestors none;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
120	HTTP Header	Fehlender Content-Security-Policy Header "plugin-types"	Der HTTP-Content-Security-Policy Header ermöglicht es dem Website-Administrator, Ressourcen zu verwalten, die der Browser für eine bestimmte Seite laden darf. Mit wenigen Ausnahmen beinhalten die Richtlinien meist die Angabe von Server- und Skript-Endpunkten. Dies hilft gegen Cross-Site-Scripting-Angriffe (XSS). Mittels "plugin-types" werden MIME-types definiert, die in der Web-Applikation erlaubt sind. Alle anderen MIME-Types werden in einer Sitzung mit dem Webserver abgelehnt. Ohne diese Direktive wird es einem Angreifer erlaubt, nach Code-Einschleusung von einer unsicheren Quelle Inhalte nachzuladen und somit an sensible Informationen der aktuellen Sitzung zu gelangen, bzw. weitere Maleware in der bestehenden Sitzung zu laden.	Es sollte der Content-Security-Policy Header "plugin-types" gesetzt werden. Für den lighttpd-Server kann dies mittels der folgenden Einstellung erfolgen: ~~~ bash Header set Content-Security-Policy "plugin-types application/pdf;" # Definieren der Header-Variablen var.common-response-headers += ("Content-Security-Policy" => "plugin-types application/pdf;") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							
121	HTTP Header	HTTP Strict Transport Security	HTTP Strict Transport Security (HSTS) ist ein Sicherheitsmechanismus für HTTPS-Verbindungen, der sowohl vor Aushebelung der Verbindungsverschlüsselung durch eine Downgrade-Attacke, als auch vor Session Hijacking schützen soll. Hierzu kann ein Server mittels des HSTS-Headers dem Browser des Anwenders mitteilen, in Zukunft für eine definierte Zeit (max-age) ausschließlich verschlüsselte Verbindungen zu dieser Domain zu nutzen.	Es wird empfohlen, den HSTS-Header zu setzen. Lighttpd-Server lässt sich mittels folgender Konfiguration so konfigurieren, dass dieser Header immer gesetzt wird: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Strict-Transport-Security" => "max-age=63072000; includeSubdomains; preload") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	High (H)	None (N)	Required (R)	Unchanged (U)	High (H)	High (H)	None (N)							
122	HTTP Header	Kein X-Frame-Options Header	Der X-Frame-Options Header teilt dem Browser mit, ob die Webseite in einem Frame angezeigt werden darf oder nicht. Möglich sind hierfür die Werte "DENY" (Seite darf nicht in Frame angezeigt werden), "SAMEORIGIN" (Seite darf nur von Frames auf derselben Domain angezeigt werden) und "ALLOW-FROM" (Seite darf von spezifizierter Domäne und URL angezeigt werden). Durch das Einbinden externer iFrames kann ein Angreifer Clickjacking Attacken durchführen.	Es wird empfohlen, den X-Frame-Options Header zu setzen. Dies kann beim lighttpd-Server über die folgende Konfiguration erfolgen: ~~~ bash # Definieren der Header-Variablen var.common-response-headers += ("Strict-Transport-Security" => "max-age=63072000; includeSubdomains; preload") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header = var.common-response-headers ~~~~	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	None (N)	Low (L)	None (N)							
123	Konfiguration	SSH mit Passwort Authentifizierung	Der SSH-Dienst erlaubt die Authentifizierung mittels Passwort. Ein Angreifer könnte durch ein Brute-Force Angriff das Passwort erraten. *Info:*[https://github.com/dev-sec/ssl-baseline](https://github.com/dev-sec/linux-baseline) und https://shattered.io/.	Es wird empfohlen, die SSH Authentifizierung ausschließlich mit Public/Private-Keys zu erlauben. Dies kann unter Linux mit dem folgenden Eintrag in der ssh-Konfigurationsdatei erfolgen: ~~~ bash PasswortAuthentication off ~~~~ *Info:* https://github.com/dev-sec/chef-ssh-hardening	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	None (N)	Unchanged (U)	Low (L)	Low (L)	None (N)							
124	Verschlüsselung	Kein TLS Fallback Signaling Cipher Suite Value (SCSV)	Der Dienst auf Port 443 stellt einen HTTPS-Server mit TLS-Verschlüsselung zur Verfügung. Dieser Dienst ist so konfiguriert, dass ein Angreifer mit einem Man-in-the-Middle-Angriff zwischen die beiden Verbindungspartner schalten und die TLS-Anfrage des Clients so umformen kann, dass anstatt TLS 1.2 ein schwächeres Verfahren, wie TLS 1.0, verwendet wird. Auf diese Weise wird die Qualität der Verschlüsselung geschwächt und es wird dem Angreifer vereinfacht die Daten zu entschlüsseln.	Es wird dringend empfohlen, den Web-Server so anzupassen, dass dieser SCSV unterstützt, um die Kompromittierung der verschlüsselten Verbindung zu vermeiden. Dazu müssen alle SSL/TLS Protokolle bis auf TLSv1.2 deaktiviert werden.	192.168.2.65	gering	Network (N)	High (H)	None (N)	Required (R)	Unchanged (U)	Low (L)	None (N)	None (N)							
125	Konfiguration	ICMP Timestamp Request Remote Date Disclosure	Das Senden von ICMP-Informationen, wie dem Zeitstempel / Timestamp des Systems, erlaubt es einem Angreifer, Informationen über das Zielsystem zu erhalten, welche er für weitere, zielgerichtete Angriffe nutzen kann. CVE-1999-0524	Es wird empfohlen, ICMP-Timestamps zu deaktivieren. Dies kann in Linux mit dem folgenden Eintrag in die sysctl.conf erfolgen: ~~~ bash net.ipv4.tcp_timestamps = 0 ~~~~	192.168.2.65	gering	Network (N)	High (H)	None (N)	None (N)	Unchanged (U)	Low (L)	None (N)	None (N)							
126	Fingerprint	lighttpd Server Banner	Der Web-Server (lighttpd/1.4.31) gibt seine Version aus. Diese Informationen kann ein Angreifer für weitere gezielte Angriffe verwenden. *Request:* ~~~ http GET /addons/mh/js/cloudmatic.js HTTP/1.1 Host: 192.168.2.65 Accept-Encoding: gzip, deflate Accept: */* Accept-Language: en User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0) Connection: close Referer: http://192.168.2.65/addons/mh/index.cgi ~~~~ *Response:* ~~~ http HTTP/1.1 200 OK Content-Type: text/javascript Accept-Ranges: bytes ETag: "4151197402" Last-Modified: Mon, 21 Aug 2017 12:07:43 GMT Content-Length: 3390 Connection: close Date: Tue, 20 Feb 2018 16:36:03 GMT Server: lighttpd/1.4.31 $(document).ready(function() { ... ~~~~	Es wird empfohlen, alle HTTP-Header, die auf die Version der eingesetzten Software hinweisen, zu entfernen. Damit wird es einem Angreifer erschwert, zielgerichteter Angriffe gegen den Web-Server zu starten und er benötigt mehr Zeit, um den Web-Server evtl. zu kompromittieren. Im lighttpd-Server kann dies wie folgt konfiguriert werden ~~~ bash server.tag = "" ~~~~	192.168.2.65	gering	Network (N)	High (H)	None (N)	None (N)	Unchanged (U)	Low (L)	None (N)	None (N)							
127	HTTP Header	Kein X-XSS-Protection Header	Der HTTP X-XSS-Protection Header ist eine Funktion vom Internet Explorer, Chrome, Firefox und Safari, die das Laden von Seiten verhindert, wenn der Browser Cross-Site-Scripting-Angriffe (XSS) erkennt. Durch diese Funktion kann vielen XSS-Angriffen vorgebeugt werden.	Es wird empfohlen, den X-XSS-Protection: 1; mode=block zu setzen. Dies kann bei lighttpd über die folgenden Einträge in der lokalen Konfiguration erfolgen: ~~~ bash # Definieren der Header-Variablen var.security-response-headers += ("X-XSS-Protection" => "1; mode=block") # Setzen des Response-Headers im entsprechenden Scope des Servers setenv.add-response-header += var.security-response-headers ~~~~ *Info:*https://geekflare.com/http-header-implementation/	192.168.2.65	mittel	Network (N)	Low (L)	None (N)	Required (R)	Unchanged (U)	Low (L)	Low (L)	None (N)							

128	Fingerprint	lighttpd Server ETag Header	Der Web-Server ist durch eine Schwachstelle bezüglich der Offenlegung von Informationen betroffen, d.h. der ETag-Header gibt sensible Informationen preis. Ein Angreifer könnte somit an die Inode Nummer für die angeforderten Dateien gelangen. *Request:* ~~~ http GET /addons/mh/js/cloudmatic.js HTTP/1.1 Host: 192.168.2.65 Accept-Encoding: gzip, deflate Accept: */* Accept-Language: en User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0) Connection: close Referer: http://192.168.2.65/addons/mh/index.cgi ~~~~ *Response:* ~~~ http HTTP/1.1 200 OK Content-Type: text/javascript Accept-Ranges: bytes ETag: "4151197402" Last-Modified: Mon, 21 Aug 2017 12:07:43 GMT Content-Length: 3390 Connection: close Date: Tue, 20 Feb 2018 16:36:03 GMT Server: lighttpd/1.4.31 $(document).ready(function() { ... ~~~~	Es wird empfohlen, den Parameter __file.etags__ in der lighttpd-Konfiguration zu entfernen. Dies kann wie folgt geschehen: ~~~ bash expire.url = ("" => "modification") etag.use-inode = "disable" etag.use-mtime = "disable" etag.use-size = "disable" static-file.etags = "disable" ~~~~	192.168.2.65	gering	Network (N)	High (H)	None (N)	None (N)	Unchanged (U)	Low (L)	None (N)	None (N)							
129	Fingerprint	lighttpd Server Date Header	Der Web-Server ist durch eine Schwachstelle bezüglich der Offenlegung von Informationen betroffen, d.h. der Date-Header wird mit übermittelt. *Request:* ~~~ http GET /addons/mh/js/cloudmatic.js HTTP/1.1 Host: 192.168.2.65 Accept-Encoding: gzip, deflate Accept: */* Accept-Language: en User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0) Connection: close Referer: http://192.168.2.65/addons/mh/index.cgi ~~~~ *Response:* ~~~ http HTTP/1.1 200 OK Content-Type: text/javascript Accept-Ranges: bytes ETag: "4151197402" Last-Modified: Mon, 21 Aug 2017 12:07:43 GMT Content-Length: 3390 Connection: close Date: Tue, 20 Feb 2018 16:36:03 GMT Server: lighttpd/1.4.31 $(document).ready(function() { ... ~~~~	Es wird empfohlen, den Header in der lighttpd-Konfiguration zu entfernen. Dies kann wie folgt erzielt werden: ~~~bash setenv.set-response-header = ("Date" => " ") ~~~~	192.168.2.65	gering	Network (N)	High (H)	None (N)	None (N)	Unchanged (U)	Low (L)	None (N)	None (N)							
130	Fingerprint	lighttpd Server Last-Modified Header	Der Web-Server ist durch eine Schwachstelle bezüglich der Offenlegung von Informationen betroffen, d.h. der Last-Modified-Header wird mit übermittelt. *Request:* ~~~ http GET /addons/mh/js/cloudmatic.js HTTP/1.1 Host: 192.168.2.65 Accept-Encoding: gzip, deflate Accept: */* Accept-Language: en User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0) Connection: close Referer: http://192.168.2.65/addons/mh/index.cgi ~~~~ *Response:* ~~~ http HTTP/1.1 200 OK Content-Type: text/javascript Accept-Ranges: bytes ETag: "4151197402" Last-Modified: Mon, 21 Aug 2017 12:07:43 GMT Content-Length: 3390 Connection: close Date: Tue, 20 Feb 2018 16:36:03 GMT Server: lighttpd/1.4.31 $(document).ready(function() { ... ~~~~	Es wird empfohlen, den Header in der lighttpd-Server zu entfernen. Dies kann durch die folgende Einstellung in der Konfiguration erfolgen: ~~~ bash setenv.set-response-header = ("Last-Modified" => " ") ~~~	192.168.2.65	gering	Network (N)	High (H)	None (N)	None (N)	Unchanged (U)	Low (L)	None (N)	None (N)							
131	Verschlüsselung	Schwache SSL/TLS Algorithmen	Der Web-Server bietet schwache Verschlüsselungsalgorithmen wie RC4, DES, 3DES, CBC sowie MD5 und SHA1 Hashalgorithmen an. Diese gelten als unsicher und sollten nicht mehr verwendet werden, da ein Angreifer die Kommunikation entschlüsseln kann, z. B.: ~~~ bash TLS_ECDHE_RSA_WITH_RC4_128_SHA TLS_RSA_WITH_RC4_128_MD5 TLS_RSA_WITH_SEED_CBC_SHA TLS_RSA_WITH_RC4_128_SHA TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS_DHE_RSA_WITH_DES_CBC_SHA TLS_RSA_WITH_DES_CBC_SHA ~~~	Es wird empfohlen, ausschließlich eine starke Verschlüsselung zu nutzen, wie z. B. die folgenden Algorithmen: ~~~ bash ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 ECDHE_RSA_WITH_AES_128_GCM_SHA256 ECDHE_RSA_AES256_SHA384 ECDHE_RSA_AES128_SHA256 ECDHE-ECDSA-CHACHA20-POLY1305 ECDHE-ECDSA-AES256-SHA384 ECDHE-ECDSA-AES256-GCM-SHA384 ECDHE-ECDSA-AES128-SHA256 ECDHE-ECDSA-AES128-GCM-SHA256 ~~~ Es sollten keine DES, 3DES, RC4 und CBC Algorithmen und keine MD5 sowie SHA1 Hashalgorithmen mehr für die TLS-Verschlüsselung verwendet werden. *Info:* https://github.com/dev-sec/ssl-baseline	192.168.2.65	mittel	Physical (P)	Low (L)	None (N)	None (N)	Unchanged (U)	High (H)	High (H)	Low (L)							
132	Verschlüsselung	Schwache SSL/TLS Protokolle	Der Server verschlüsselt die Kommunikationen mit SSLv3 und TLSv1.0/TLSv1.1. Diese TLS-Versionen sind durch mehrere kryptographische Fehler (POODLE, DROWN, BEAST, CBC-Implementierung) betroffen. Ein Angreifer kann diese Fehler ausnutzen, um Man-in-the-Middle-Angriffe durchzuführen oder die Kommunikation zwischen dem betroffenen Dienst und den Clients zu entschlüsseln sowie Denial-of-Service Angriffe durchzuführen.	Es sollten für verschlüsselte Kommunikationen ausschließlich TLSv1.2 genutzt werden.	192.168.2.65	hoch	Network (N)	Low (L)	None (N)	None (N)	Unchanged (U)	Low (L)	Low (L)	Low (L)							
133	Verschlüsselung	Schwache SSH Kex Algorithmen	Der SSH-Dienst verwendet schwache Algorithmen (z.B. diffie-hellman-group14-sha1) für den Schlüsselaustausch. Diese Algorithmen gelten als nicht mehr sicher, da ein Angreifer diese in vertretbarem Aufwand brechen kann. ~~~ bash ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1 ~~~ *Info:* https://github.com/dev-sec/ssh-baseline und http://shattered.io/	Es wird empfohlen, starke Kex-Algorithmen zu verwenden, z. B.: ~~~ bash curve25519-sha256@libssh.org diffie-hellman-group-exchange-sha256 ~~~	192.168.2.65	mittel	Network (N)	High (H)	None (N)	Required (R)	Unchanged (U)	High (H)	High (H)	None (N)							
134	Verschlüsselung	Schwache SSH Message Authentication Code Algorithmen	Der SSH-Dienst verwendet nicht alle starken MAC-Algorithmen. ~~~ bash hmac-sha2-256,hmac-sha2-512 ~~~ *Info:* https://github.com/dev-sec/ssh-baseline und http://shattered.io/	Es wird empfohlen, stake MAC-Algorithmen zu verwenden. z.B. ~~~ bash hmac-sha2-512-etm@openssh.com hmac-sha2-256-etm@openssh.com hmac-sha2-512 hmac-sha2-256 ~~~	192.168.2.65	gering	Network (N)	High (H)	None (N)	Required (R)	Unchanged (U)	None (N)	Low (L)	None (N)							

<table>
<tr><td>135</td><td>Verschlüsselung</td><td>Schwache SSH Verschlüsselungsalgorithmen</td><td>Der SSH-Dienst unterstützt nicht alle starken Verschlüsselungsalgorithmen.

~~~ bash
aes256-ctr,aes192-ctr,aes128-ctr
~~~~

Info: https://github.com/dev-sec/ssh-baseline und http://shattered.io/</td><td>Es wird empfohlen, starke Verschlüsselungsalgorithmen zu nutzen, z. B.:

~~~ bash
chacha20-poly1305@openssh.com
aes256-gcm@openssh.com
aes128-gcm@openssh.com
aes256-ctr
aes192-ctr
aes128-ctr
~~~~</td><td>192.168.2.65</td><td>gering</td><td>Network (N)</td><td>High (H)</td><td>None (N)</td><td>Required (R)</td><td>Unchanged (U)</td><td>Low (L)</td><td>None (N)</td><td>None (N)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>136</td><td>Web-Anwendung</td><td>Aufruf externer Dienste (DNS und HTTP) durch die Web-Anwendung</td><td>Bei einem externen Dienstaufruf wird die Web-Anwendung dazu veranlasst einen externen Web-Server, Mail-Server, FTP-Server oder DNS-Server aufzurufen, um von diesem Inhalte abzurufen. Auf diese Weise können Informationen an diesen Dienst übermittelt, oder Informationen von diesem Dienst abgerufen werden. Dies kann durch einen Angreifer auch dazu ausgenutzt werden, Systeme Dritter anzugreifen.

Request:

~~~ http
POST / HTTP/1.1
Host: 192.168.2.65:2010
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) QupZilla/2.2.5 Chrome/61.0.3163.140 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
DNT: 1
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.8
Connection: close
Content-Length: 98

<?xml version=""1.0""?><!DOCTYPE methodcall PUBLIC ""-//B/A/EN"" ""http://www.targetsystem.victim""><methodCall><methodName>system.listMethods</methodName><params/></methodCall>
~~~~

Response:

~~~ http
HTTP/1.1 200 OK
Content-Length: 0
Connection: close
~~~~</td><td>Es wird dringend empfohlen, Eingabewerte, die vom Browser gesendet werden, zu prüfen und die Aufrufe von externen Diensten aus der Web-Anwendung heraus auf das Nötige zu beschränken.</td><td>192.168.2.65</td><td>hoch</td><td>Network (N)</td><td>Low (L)</td><td>None (N)</td><td>None (N)</td><td>Unchanged (U)</td><td>High (H)</td><td>None (N)</td><td>None (N)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>137</td><td>HTTP Header</td><td>Kein Anti-CSRF Token</td><td>Bei einem Cross-Site-Forgery Angriff wird ein Opfer dazu gebracht unbewusst eine HTTP-Anfrage an ein Ziel zu senden, um unbewusst eine Aktion auf diesem durchzuführen. Die zugrunde liegende Ursache ist die Anwendungsfunktionalität der Webseite, mit vorhersagbaren URL / Form-Aktionen in einer wiederholbaren Weise. Cross-site-request-forgery ist auch bekannt als CSRF, XSRF, One-Click-Angriff, Session Riding und Sea Surf. Durch das Mitsenden eines eindeutigen, individuellen Anti-CSRF-Tokens, z. B. in einem Hidden-Field der Webseite, werden diese Angriffe erschwert.

Request:

~~~ http
GET /addons/mh/clregister.cgi HTTP/1.1
Host: 192.168.2.65
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Referer: http://192.168.2.65/addons/mh/index.cgi
~~~~

Response:

~~~ http
HTTP/1.1 200 OK
Connection: close
Date: Tue, 20 Feb 2018 16:36:03 GMT
Server: lighttpd/1.4.31
Content-Length: 13334

...
<form action="doclregister.cgi" method="GET" name="registerform" id="registerform">

<input name="cb_snr" type="hidden" id="cb_snr" value="00-1A-22-07-A1-E9" />
<input name="cb_tkey" type="hidden" id="cb_tkey" value="00-1A-22-00-05-</td><td>Es wird empfohlen, ein Anti-CSRF Token bei jedem Request zu generieren und zu prüfen.

In lighttpd-Server kann dies mittels Aktivierung des Moduls `mod_csrf` erreicht werden. Eine Konfiguration dieses Moduls kann wie folgt vorgenommen werden :

~~~ bash
server.modules += ("mod_csrf")
$HTTP["url"] =~ "^/(someurl|cgi-bin)/(.+)" {
csrf.protection = "enable"
}
csrf.cookie-lifetime = 600;
csrf.cookie-domain = "someurl";
csrf.send-cookie-name = "csrf";
csrf.receive-header-name = "X-Csrf-Token";
~~~</td><td>192.168.2.65</td><td>hoch</td><td>Network (N)</td><td>High (H)</td><td>None (N)</td><td>Required (R)</td><td>Unchanged (U)</td><td>High (H)</td><td>High (H)</td><td>Low (L)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>138</td><td>Web-Server</td><td>Content-Type nicht korrekt gesetzt</td><td>Wenn vom Web-Server der Content-Type nicht entsprechend des Dateityps gesetzt wird, wird die empfangene Datei von vielen Browsern als HTML-Datei interpretiert. Dies kann zu fehlerhaften Darstellungen oder wenn die Datei von einem Angreifer modifiziert werden kann, zur Einschleusung von Code in den Browser des Anwenders führen.

Request:

~~~ http
GET /favicon.ico HTTP/1.1
Host: 192.168.2.65
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) QupZilla/2.2.5 Chrome/61.0.3163.140 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
DNT: 1
Referer: http://192.168.2.65/index.htm?sid=@jomcQvAH0I@
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.8
Connection: close
~~~

Response:

~~~ http
HTTP/1.1 200 OK
Server: ise GmbH HTTP-Server v2.0
Accept-Ranges: bytes
Cache-Control: no-store, no-cache
Content-Type: text/html; charset=iso-8859-1
Content-Length: 1406
Date: Tue, 20 Feb 2018 18:03:52 GMT
Last-Modified: Mon, 21 Aug 2017 13:10:50 GMT
Connection: close

.............h.......(.......
..............................===.AAA.HHH.JJJ.LLL.MMM.RRR.TTT.XXX.```.ccc.lll.qqq.uuu.vvv.www.yyy.zzz.|||.}}} ....
....
....
....</td><td>Es wird empfohlen, bei jeder Datei den entsprechenden Dateityp / Inhalt durch den Web-Server, bzw. die Web-Anwendung zu setzen, um Fehlerquellen und Angriffsfläche in der Web-Anwendung zu vermindern.</td><td>192.168.2.65</td><td>mittel</td><td>Network (N)</td><td>High (H)</td><td>High (H)</td><td>Required (R)</td><td>Unchanged (U)</td><td>High (H)</td><td>High (H)</td><td>Low (L)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>139</td><td>Web-Anwendung</td><td>Unverschlüsselte Authentifizierung und Kommunikation</td><td>Die Web-Anwendung erlaubt eine unverschlüsselte Übertragung von Benutzername und Passwort. Ein Angreifer könnte durch einen MitM-Angriff an sensible Informationen gelangen.</td><td>Es wird empfohlen, sensible Informationen, wie Benutzername und Passwort, immer über eine verschlüsselte Verbindung zu übertragen.</td><td>192.168.2.65 (tcp/1999)
192.168.2.65 (tcp/2001)
192.168.2.65 (tcp/2010)</td><td>hoch</td><td>Network (N)</td><td>High (H)</td><td>None (N)</td><td>Required (R)</td><td>Unchanged (U)</td><td>High (H)</td><td>High (H)</td><td>Low (L)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>140</td><td>Web-Anwendung</td><td>Zugriff auf XML-HTTP-RPC ohne Authentifizierung</td><td>Der Zugriff auf die HTTP-basierten XML-RPC Schnittstelle ist ohne Authentifizierung aus dem kompletten Netzwerk-Segment möglich. Dies erlaubt es einem Angreifer, sensible Daten abzufragen und zu manipulieren. Des Weiteren kann ein Angreifer alle angeschlossenen Geräte kontrollieren und steuern.

Info: [CVE-2018-7301](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7301) und http://atomic111.github.io/article/homematic-ccu2-xml-rpc</td><td>Es wird empfohlen, sensible Informationen ausschließlich authentifizierten Benutzern zugänglich zu machen.</td><td>192.168.2.65 (tcp/1999)
192.168.2.65 (tcp/2001)
192.168.2.65 (tcp/2010)</td><td>sehr hoch</td><td>Network (N)</td><td>Low (L)</td><td>None (N)</td><td>None (N)</td><td>Unchanged (U)</td><td>High (H)</td><td>High (H)</td><td>High (H)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>141</td><td>Web-Anwendung</td><td>Remote Code Execution</td><td>Der TCL-Script Interpreter ist nicht über ein Session Handling abgesichert, d. h. ein nicht authentifizierter Angreifer kann über den TCL-Script Interpreter Schadcode auf dem System ausführen. Dies kann über einen einfachen POST-Request über den /Test.exe Pfad erfolgen.

Info: [CVE-2018-7297](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7297) und http://atomic111.github.io/article/homematic-ccu2-remote-code-execution</td><td>Es wird empfohlen, den TCL-Script Interpreter ausschließlich über eine authentifizierte Sitzung zu benutzen.</td><td>192.168.2.65</td><td>sehr hoch</td><td>Network (N)</td><td>Low (L)</td><td>None (N)</td><td>None (N)</td><td>Unchanged (U)</td><td>High (H)</td><td>High (H)</td><td>High (H)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>