Automatiserad budgivning på internet

Hur fungerar det och hur går det att skydda sig?

LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Datateknik

Examensarbete:
André Ahlfors Dahl

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2011

Sammanfattning

Detta examensarbete beskriver hur mjukvara för automatiserad budgivning på t.ex. auktionssajter skulle kunna fungera. Examensarbetet är uppdelat i tre delar. De olika delarna består av:

- Insamling av auktionsdata som sker genom att läsa av hemsidan som auktionen hålls på och skicka vidare obehandlad data till huvudapplikationen för vidare bearbetning.

- En grafisk presentation av auktionsdata som visas uppdelade i individuella auktioner i det egenutvecklade gränssnittet efter att först ha behandlat data som blev inläst från hemsidan.

- Tekniker som kan skydda mot automatisering av budgivning behandlades också. Resultatet är att inte en enskild lösning fungerar bäst utan det är kombinationen av flera. Det är detta som gör det mycket svårare för en angripare att lyckas med att ta fram en fungerande mjukvara för automatiserad budgivning.

Resultatet för applikationen blev som förväntat en fungerande budgivningsrobot som på ett väldigt enkelt sätt för användaren visar hur budgivningen sker. Motåtgärder som ska förhindra den här typen av applikationer togs också fram och fungerade väl mot den egenutvecklade applikationen. Då det finns många olika sätt att kringgå skydden är det svårt, om inte omöjligt, att ta fram en universell lösning som alltid fungerar.

Nyckelord: Budgivning, Budvakt, Autobud, Auktion, Auktionsscanner, Budrobot

Abstract

This thesis describes how software for automated bidding on for example auction sites could function. The thesis is divided into three parts. The elements are composed of:

• Collecting bid data from the website that the auction is held on and pass the raw data to the main application for further processing.

• A graphical presentation of the auction data is shown divided into individual auctions in the proprietary interface after having first processed the data that was collected from the website.

• Techniques that can help protect against automated bidding is also discussed and the result is not that a single solution works best, but the combination out of several techniques that make it much harder for an attacker to succeed in producing a working software for automated bidding.

The result of the application was as expected, a functioning robot that makes bidding a very simple task, showing how the bidding takes place in real time. Countermeasures to prevent this type of application was also designed and worked well against the proprietary application. When there are so many different ways to evade the imposed security settings, it is difficult, if not impossible, to develop a universal solution that always works.

Keywords: Bidding, Bid watch, Auto bidding, Auction, Auction Scanner, Bid robot

Förord

Jag vill tacka min handledare och examinator som har kommit med många bra tips och idéer på arbetet.

Jag vill tacka min familj och mina närmaste vänner för att de stöttade och hjälpte mig uppför den stora uppförsbacken på slutet av arbetet. Tack för att ni stod ut när jag gömde mig i garderoben den sista tiden för att få allt på plats.

André Ahlfors Dahl
2011-06-12

Innehållsförteckning

1 Inledning

Dagens samhälle blir allt mer digitaliserat. Detta har medfört att vi gör fler saker över internet än någonsin tidigare. En av dessa aktiviteter är budgivning på internet. Hur rättvist är det med budgivning över internet? Kan vi ha samma garantier att det finns någon människa bakom tangentbordet som att det finns någon bakom skylten i auktionskammaren? Problemet ligger alltså i att försöka hindra att få automatiska budgivningsrobotar att fungera mot den här typen av tjänster.

Varför är automatisk budgivning dåligt? Vissa auktionssajter tillhandahåller en fullt legitim sådan funktion på vissa av de enskilda auktionerna, men tjänsteleverantören vill inte erbjuda det till alla auktioner. Detta för att skapa en mer rättvis budgivningschans mot de som inte kan ladda sitt budkonto med väldigt mycket krediter och kan då ”set and forget” och räkna med att de vinner auktionen. Genom att begränsa automatiken så ges fler budgivare chansen att vinna varan då det är live budgivning som gäller. Detta innebär att du måste sitta och vakta auktionen i, till exempel trettio sekunders intervall för att inte riskera förlora varan. Detta tror jag är en strategi för att framförallt få in nya budgivare i en auktion. När det är en live auktion innebär detta också att många bud kan läggas under tiden auktionen pågår. Detta skulle kunna innebära en högre andel bud på kort tid än vad som sker hos en motsvarande auktion som tillåts ha automatisk budgivning, eftersom systemet lägger ett bud först när auktionen börjar närma sig sin utgångstid. Ett exempel är när det enbart är tre sekunder kvar så hoppar systemet in och lägger ett bud.

En liknelse till automatiserad budgivning kan ges i online-pokervärlden där förekomsten av automatiska robotar är strikt förbjudet och straffbart. En ”robotspelare” här skulle kunna spela hem mycket stora pengar då den har bättre förutsättningar. Detta för att den kan följa spelet, räkna på statistik och optimera spelgången efter spelets gång, egenskaper som enbart de bästa inom sporten anses kunna. Då det handlar om stora mänger pengar så finns det stora krafter åt båda hållen som inte drar sig för utveckling av sådan programvara.

Det är också den enskilt största anledningen till varför det är mycket svårt att hitta tidigare arbeten som behandlar ämnet. Detta just för att det är lite i en gråzon enligt lag att utveckla den här typen av mjukvara, och de som lyckas brukar inte publicera sina resultat för allmänheten när det finns pengar att tjäna på att sälja resultaten.

1.1 Mål

Målet med arbetet är ta reda på om det är möjligt att automatisera budgivning med mjukvara som tar över människans roll och agerar utifrån en uppsättning fördefinierade regler och värden. Exempel på regler och värden är totala antalet bud som får läggas eller ett maximalt pris på varan. Arbetet kommer också att undersöka och ge förslag på hur det går att förhindra den här typen av mjukvara från att fungera. Detta leder fram till följande två fundamentala frågeställningar:

- Går det att utveckla en applikation för automatiserad budgivning?
- Hur kan man skydda sig mot automatiserad budgivning?

För att bryta ner frågeställningen i hur automatiserad budgivning skulle kunna fungera så har följande frågeställningar tagits fram:

- Vad är auktionsdata?
- Vilka data utgör en specifik auktion?
- Hur kan vi identifiera de olika delarna?

Dessa frågeställningar som är avgörande för att skapa en applikation som ska utföra automatiserad budgivning besvaras i kapitel fyra.

1.2 Innehåll

- En genomgång av vilken typ av data som ska samlas in och hur.
- En genomgång av hur utvecklingen av testverktyg genomfördes.
- En genomgång av skyddsåtgärder som förhindrar automatisk budgivning.
- Idéer på vad som kan implementeras i testverktyget i framtiden.

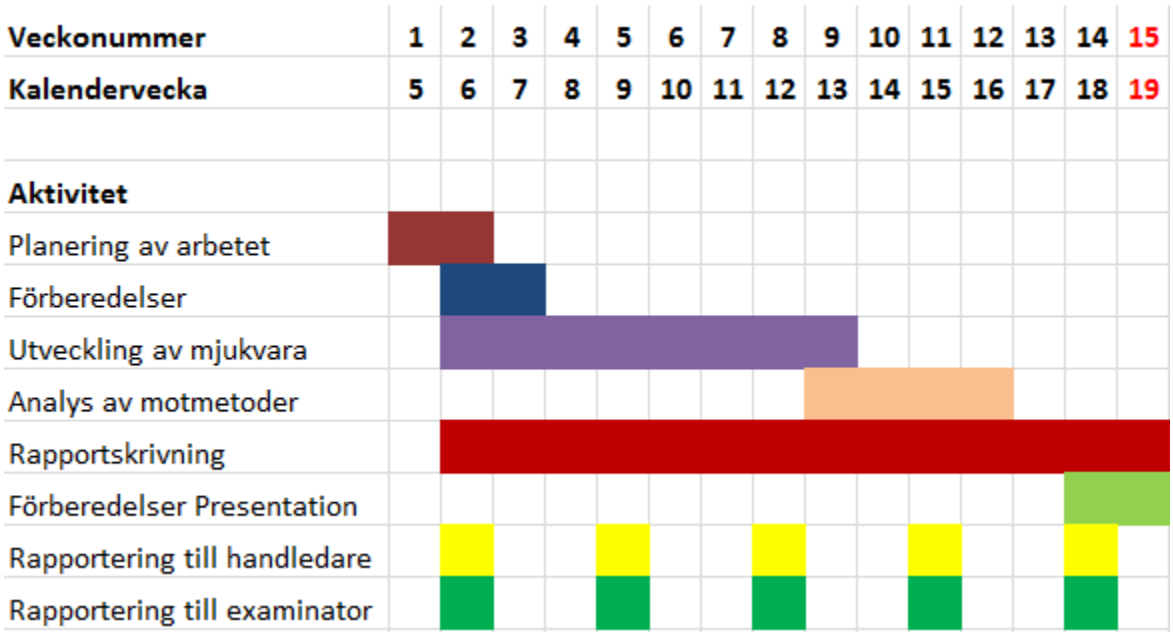
2 Metod, faser och källor

2.1 Metod och faser

Den arbetsmetod som har använts vid arbetet har bestått av att dela upp arbetet i olika faser. De tre huvudfaserna är följande:

- Förberedelse.
- Utveckling av mjukvara.
- Analys av skyddsåtgärder.

Dessa faser föregicks utav två mindre intensiva aktiviteter kallade planering och förberedelser. Rapportskrivningen, som är en stor del av arbetet, gjordes kontinuerligt under hela arbetet för att minska arbetsbördan på slutet. Hela fördelningen inklusive en tidfördelning representeras av följande Gantschema:



Figur 2:1 Gantschema.

Bilden visar tidsplaneringen för examensarbetet.

2.2 Källgranskning

I denna rapport har det eftersträvats att använda källor som är så nära upphovsmännen som möjligt. Detta då dessa källor är mycket trovärdiga just för att de är upphovsmakarens men också för att de brukar innehålla den mest uppdaterade och korrekta informationen inom ämnet. I de fall ingen enskild

upphovsman/organisation kunnat bli identifierad och hänvisad till, så har uppslagsverket Britannica[1] använts.

3 Teknisk bakgrund

För att till fullo kunna tillgodogöra sig innehållet i detta examensarbete krävs utöver grundläggande datorvana även en generell kompetens inom följande områden:

- HTML.
- CSS.
- PHP.
- JSON.
- C#.
- Wireshark.

Dessa ämnen förklaras nedan lite djupare för de som är intresserade eller behöver en snabb genomgång.

3.1 HTML

HTML är en förkortning som står för Hypertext Markup Language. HTML är en standard över hur websidor är strukturerade i sina grundläggande element. HTML är alltså ett språk som beskriver för en enhet hur websidan ska läsas och struktureras. HTML ger möjligheter att:

- Inhämta annan online information genom att klicka på hyperlänkar.
- Publicera online dokument (hemsida) med rubriker, text, bilder, tabeller osv.
- Skapa och designa formulär som kan användas till att både inhämta och presentera data.

[1] Encyclopædia: Britannica Online Encyclopedia (Läst 2011-06-11)
[2] För en fullständig lista över alla giltiga namn se: World Wide Web Consortium: *HTML/Element.* (Läst 2011-

- Till exempel inkludera ljud och video direkt på hemsidan utan att använda sig utav hyperlänkar.

Språket i HTML-filer, vilka oftast har filändelsen .html eller .htm, är uppbyggt av taggar som kapslar in text. En sådan inkapsling av data utgör ett element. Det är med hjälp av taggarna som till exempel webbläsare vet hur den ska presentera websidan för användaren. En tagg består av <namn på tag[2]> och de flesta sådana taggar måste avslutas med </namn på tag>. De taggar som inte behövs avslutas på ett sådant sätt kallas tomma element.

Exempel på hur en enkel hemsida i HTML kodning ser ut, skulle kunna vara såhär:

```
<!doctype html>
<html lang="sv">
<head>
  <title>Detta är en titel på hemsidan</title>
</head>
<body>
  <p>Här är ett stycke text på hemsidan</p>
</body>
</html>
```

Den senaste versionen av HTML är version 5 som dock ännu inte är helt klar. Den befinner sig som ett utkast, dock är sista dag för förändring av detta utkast 3 augusti 2011[3]. Tidigare versioner och när dessa blev klara är:

Tabell 3:1 Versionshistorik för olika utgåvor av HTML inklusive årtal.[4]

Version	Årtal
HTML+	1993
HTML2.0	1995
HTML3.2	1997
HTML4.01	1999

3.2 CSS

CSS är en förkortning som står för Cascading Style Sheets. CSS är ett språk som beskriver layouten, det vill säga presentationen av en hemsida. Detta

[2] För en fullständig lista över alla giltiga namn se: World Wide Web Consortium: *HTML/Element.* (Läst 2011-06-11.)
[3] World Wide Web Consortium: *HTML5*. (Läst 2011-06-11.)
[4] World Wide Web Consortium: *What is HTML.* (Läst 2011-06-11.)

innefattar färger, teckensnitt och layout. CSS kan hjälpa de som bygger hemsidor att skapa olika layouter för olika typer av skärmar. Ett exempel är att på vanliga skärmar presenteras hemsidan på ett sätt medan mobilen presenterar samma sida på ett annat sätt. CSS innebär att webbdesigners lättare kan anpassa sidor till olika typer av enheter utan att förända innehållet. CSS kan användas till alla typer av sidor som bygger på XML utöver HTML. Språket i CSS filer som oftast har filändelsen .css är uppbyggt enligt en uppsättning regler[5] som lättast följer av att en syntax har en eller flera egenskaper som alla har ett värde. Detta kan sammanfattas med ett exempel:

```
body {background-color: #000000;}
p {font-family: georgia, "Times New Roman", serif; font-size: 14px;}
```

CSS har en relativt kort historia som började vid sin första version (CSS 1) 1996. Nästa version blev CSS 2 som blev klar 1998. Dagens version kallat CSS 3 finns inte dokumenterad i ett stort dokument som de tidigare versionerna, utan är uppdelad i små moduler. Det finns idag den 11 juni 2011 över 50 moduler[6]. På grund av uppdelningen så har modulerna olika utvecklingsstatus och prioriteringar.

3.3 PHP

PHP är en förkortning som står för PHP: Hypertext Preprocessor. PHP är ett scriptspråk som bäddas in i vanliga html taggar för att generera dynamiska element i websidor. Mycket av syntaxen är lånad ifrån C, Java och Perl men det finns också en del php specifika funktioner[7]. PHP Script är script som läggs in i vanliga html taggar. PHP är också öppen källkod under deras egna php licens[8]. Språket i PHP-dokument, vars filnamn oftast slutar på .php, bygger på en inledande markör som definierar start (<?php) och en avslutande (?>). Ett exempel på hur php-syntax ser ut är följande:

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

Då php-koden tolkas på servern så blir det inte samma kod som visas för användaren utan det är den ”översatta” sidan:

[5] World Wide Web Consortium: *Syntax and basic data types.* (Läst 2011-06-11.)
[6] World Wide Web Consortium: *CSS Specifications.* (Läst 2011-06-11.)
[7] Hypertext Preprocessor: *General Information.* (Läst 2011-06-11.)
[8] Hypertext Preprocessor: *Licensing.* (Läst 2011-06-11.)

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <p>Hello World</p>
 </body>
</html>
```

Den senaste versionen av PHP är 5.3.6 (2011-03-10). Tidigare versioner[9] har varit:

Tabell 3:2 Versionshistorik för olika utgåvor av PHP inklusive årtal.

Version	Årtal
PHP 1.0	1995
PHP 2.0	1997
PHP 3.0	1998
PHP 4.0	2000
PHP 5.0	2004

3.4 JSON

JSON är en förkortning som står för JavaScript Object Notation. JSON är ett format för utväxling av data. Det är ett väldigt skriv- och läsvänligt format för människan. Det är också väldigt enkelt för datorer att skapa och läsa JSON baserad data. JSON har blivit inspirerat av JavaScript för att representera enkla data strukturer och objekt. Namnet till trots så är JSON ett textbaserat format som är helt språkoberoende. Det finns moduler som kan tolka JSON till över 40[10] olika programmeringsspråk och script. JSON bygger på två typer av generella data strukturer:

- En samling av par bestående av nyckel och värde, inom programmering vanligen kallat objekt.
- En ordnad lista av värden, inom programmeringen vanligen kallat array.

Dessa två strukturer kan sedan uttryckas i JSON på något av följande sätt:

- Objekt - Osorterad lista bestående av nyckel:värde par separerade med komma och inneslutna i { }.

[9] Hypertext Preprocessor: *History of PHP*. (Läst 2011-06-11.)
[10] JavaScript Object Notation: *Introducing JSON*. (Läst 2011-06-11.)

- Array – Ordnad lista med värden separerade med komma och inneslutna i [ ].
- Boolean – Sant eller Falskt.
- Sträng – Unicode[11] tecken inneslutna i dubbla citattecken.
- Nummer – Flyttal med dubbla decimalvärden.

Exempel på hur JSON-formaterad data kan se ut är:

```
{
     "firstName": "Kalle",
     "lastName": "Anka",
     "age": 100,
     "address":
     {
         "streetAddress": "Paradisappelvagen 13",
         "city": "Ankeborg",
         "state": "AB",
         "postalCode": "12345"
     },
     "phoneNumber":
     [
         {
           "type": "home",
           "number": "1234-567-890"
         },
         {
           "type": "fax",
           "number": "0987-654-321"
         }
     ]
 }
```

Douglas Crockford skapade domänen json.net 2002 där han publicerade information om hur JSON fungerade. I juli 2006 blev formatet specificerat officiellt som RFC 4627[12].

3.5 C#

C sharp är ett objektorienterat programmeringsspråk utvecklat av Microsoft som en del av .NET-plattformen. Det skapades för att vara modernt, enkelt och objektorienterat. C sharp är influerat av Java och C++. Exempelkod för hur en applikation kan se ut för en utskrift i ett terminal fönster är:

```
using System;
```

[11] Unicode 6.0.0: *Unicode 6.0.0.* (Läst 2011-06-11.)
[12] Network Working Grop: The application/json Media Type for JavaScript Object Notation (JSON). (Läst 2011-06-11.)

```
namespace HelloWorld
{
  class Hello
  {
    public static void Main()
    {
      Console.WriteLine("Hello World!");
    }
  }
}
```

Den senaste versionen av programmeringsspråket är version 4.0 som blev officiell i april 2010. Tidigare versioner och deras årtal återges i nedan tabell:

Tabell 3:3 Versionshistorik för olika utgåvor av C# inklusive årtal.

Version	Årtal
C# 1.0	2002
C# 1.2	2003
C# 2.0	2005
C# 3.0	2007
C# 4.0	2010

3.6 Wireshark

Wireshark[13] är en protokollanalysator som skapades av Gerald Combs 1998. Wireshark bytte namn till det senare från Ethereal år 2006 på grund av en tvist i rättigheterna till varumärket. Wireshark är öppen källkod som är släppt under GNU GPLv2[14] licensen. I den senaste versionen (1.6.0) har över 700 personer hjälp till att utveckla mjukvaran. Wireshark har stöd för de flesta protokoll men dess styrka ligger i möjligheterna att kunna sortera dem, över 105 000 olika filter finns att tillgå. Wireshark kan användas till att avlyssna, spela in och analysera alltifrån USB-trafik till det trådlösa nätverket.

3.7 Öresauktioner

Programvaran som tas fram i detta arbete riktar sig mot den typ av auktion som fungerar på sådant sätt att när en ny auktion startas så sätts priset på den varan till noll. Om ingen bjuder på varan återgår den till en senare budgivning. Däremot, om någon skulle bjuda på varan, så kan den som bjuder enbart höja priset med en fast summa på till exempel tio öre. Därefter startar en timer som räknar ner, i exempelvis sextio sekunder. Har ingen annan bjudit på varan så vinner senaste budgivaren om timern når noll. Om någon bjuder innan timern har räknat ner så startar timern om från sextio sekunder och den nya budgivaren står som senaste budgivare och priset har höjts till tjugo öre. För varje nytt bud som läggs så uppdateras budgivaren till den som lade budet och slutsumman för varan höjs med tio öre. Den typen av auktion brukar kallas ”öresauktion”, från britternas ”penny auctions”.

Anledningen till varför den här typen av auktion valdes var för att det är den lättaste typen av auktion att automatisera. Detta för att en applikation enbart ska kontrollera en räknare och beroende av den lägga ett bud eller inte enligt fördefinierade regler.

4 Insamling av auktionsdata

För att kunna utveckla en applikation som automatisk ska kunna lägga bud på online auktioner så behövs det kännedom om:

- Vilken auktion det rör sig om?
- Vilken tidsram auktionen har på sig?

[13] Wireshark: *About Wireshark.* (Läst 2011-06-12.)
[14] GNU Operating System: *GNU General Public License.* (Läst 2011-06-12.)

- Hur meddelas auktionssidan att ett bud har lagts från en specifik användare?

Detta är viktiga frågeställningar som måste besvaras för att kunna utveckla en fungerande programvara.

4.1 Identifiering

Exakt vilken data är nödvändig för att kunna lägga bud och vilken information är mer ”bra-att-veta”-information men inte nödvändig för en fungerande applikation? Den mest vitala grundläggande data som applikationen måste ha för att identifiera en auktion är auktionens identifikationskod och tidpunkten för det senast lagda budet. Därefter kommer den mer överflödiga informationen såsom namn på produkt, datum för start, senaste budgivare osv för att nämna några poster som kan finns med. Hur sker identifieringen av dessa poster? För att utgå från vårt exempel mot en specifik internetauktionssida så ser deras auktionsdata ut enligt följande:

```
jcb({"st":"1297863895","pu":"0","hp":"0","ah":"c30eb733edf6b35dc602f5f249e043
4c","ud":{},"auctions":[{"ai":"20319","at":"2","st":"0","cs":"","sd":"1297713
600","pi":"560","i":"30","ap":"IPAD","pn":"Apple iPad","pl":"apple-
ipad","pp":"4890","t":"1297863910","b":"49.78","u":"bsantes"},{"ai":"20288","
at":"2","st":"0","cs":"","sd":"1297796400","pi":"742","i":"30","ap":"CANONX",
"pn":"Canon EOS 60D + Objektiv","pl":"canon-eos-60d-
objektiv","pp":"10900","t":"1297863916","b":"16.86","u":"vccmv"},{"ai":"20355
","at":"2","st":"0","cs":"","sd":"1297850400","pi":"734","i":"105","ap":"PINK
","pn":"Dolce & Gabbana damklocka","pl":"dolce-gabbana-
damklocka","pp":"2000","t":"1297863994","b":"1.68","u":"dicosvd30"},{"ai":"20
357","at":"2","st":"0","cs":"","sd":"1297857600","pi":"309","i":"60","ap":"HM
","pn":"H&M presentkort","pl":"hm-
presentkort","pp":"2000","t":"1297863921","b":"1.59","u":"Goodiie"},{"ai":"20
399","at":"2","st":"0","cs":"","sd":"1298228400","pi":"641","i":"30","ap":"SA
MUE","pn":"Samsung 46\" LED-TV","pl":"samsung-46-led-
tv","pp":"11990","t":"29","b":"0","u":""}]}); });
```

Dessa data är hämtade genom att besöka den specifika hemsidan och titta på källkoden som utgör hemsidan. Från den koden så måste det sedan sorteras bort alla vanliga HTML taggar och oväsentlig text tills det som finns kvar kan identifieras som trolig auktionsdata. Till hjälp för detta användes två tilläggsprogramvaror till webbläsaren Firefox[15]:

- Firebug[16].
- Web Developer[17].

[15] Mozila Firefox: *Firefox 4 är här!*. (Läst 2011-06-12)
[16] Firebug – Web devolopment enolved: *Home page.* (Läst 2011-06-12)
[17] Chris Pederick: *Web Developer*. (Läst 2011-06-12)

Resultatet kan vara mycket svårt att tyda vid en första överblick men vid en närmare analys visar det sig att ett återkommande mönster återfinns. Detta består i vårt exempel av fem olika auktioner samt vår sessionsinformation från när informationen är hämtad. Data ovan inleds med sessionsinformation uppdelat i följande:

- St – Starttid för sessionen i Unix format.
- Pu – Om inloggad, Användarnamn.
- Hp – Okänt.
- Ah - Authentication Header, Unik sessionsid för användaren.
- Ud – Okänt.

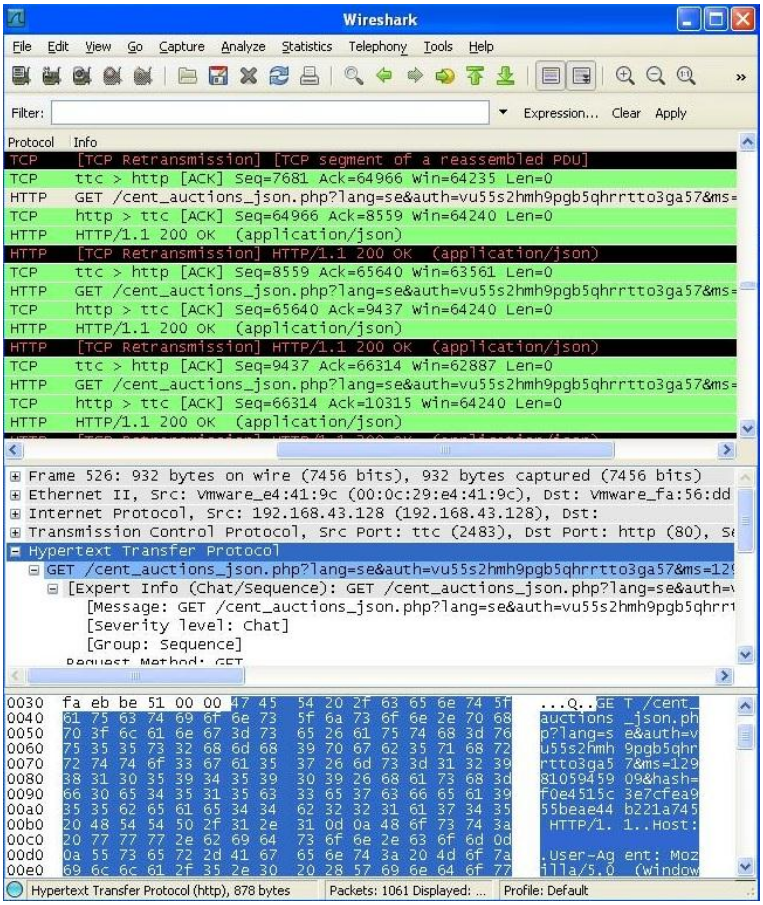
Själva auktionsposterna är uppdelade i följande:

- Ai – Unikt auktionsid.
- At – Auktionskategori.
- St – Auktionstyp.
- Cs – Okänd.
- Sd – Starttid för auktionen i Unix format.
- Pi – Okänd.
- I – Aktuellt budintervall.
- Ap – Identifieringskod för SMS.
- Pn – Läsvänlig version av produkten för människor.
- Pl – Läsvänlig version av produkten för datorer.
- Pp – Prisinformation.
- T – Senaste lagda bud i Unix format.

- B – Aktuellt bud.

- U – Budgivare för det aktuella budet.

Som synes ovan är vissa poster okända. Detta därför att själva identifieringen av posterna har skett helt på manuell väg, vilket då gjort att vissa inte har lyckats bli identifierade. Själva identifieringen skedde enligt ett flertal tester där källkodsdata och datatrafiken jämfördes till och från hemsidan mot den grafiska representationen av data. För att kunna se vilken typ av trafik som genererades så användes nätverksniffern Wireshark[18] tillsammans med Firebug. Detta gav en exakt återspegling av trafiken som flödade samt hjälp till vart i koden dessa förändringar skedde.



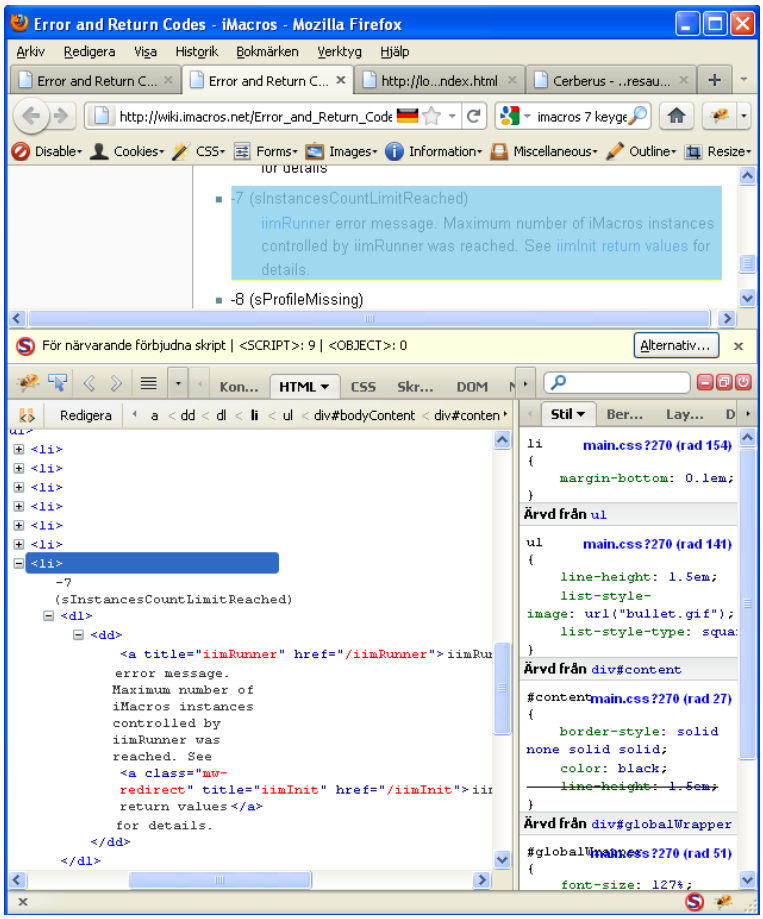
Figur 4:1 Protokollanalysatorn Wireshark.

Bilden visar hur Wireshark håller på att fånga in nätverkstrafiken mellan den lokala klienten och en webbserver.

Ovan visar en bild ifrån verktyget Wireshark som håller på att fånga in trafik mellan klient och server. Det som skall läggas märke till är att i http-trafiken

[18] Wireshark: *Home page.* (Läst 2011-06-12)

görs en automatisk ”get” med hjälp av json[19] mot målsidan. Denna målsida är en php[20]-sida där det skickas med ett sessionsid kallat ”*auth*”, en tidstämpling i Unix format kallat ”*ms*” samt ett hash-värde kallat ”*hash*”.



Figur 4:2 Firefox tillägget Firebug.

Bilden visar hur verktyget har markerat ett element på den besökta hemsidan.

Ovan är en bild på verktyget Firebug som alltså är ett tillägg till webbläsaren Firefox. Detta verktyg hjälper användaren med att automatiskt identifiera de olika element som finns på en webbsida och hitta detta elements kod i källkoden för webbsidan. Detta fungerar genom att det enda användaren behöver göra är att följa med musen på ett element på sidan så visas dess motsvarighet upp i kodfönstret. Exemplet ovan visar ett sådant scenario.

Sammanfattningsvis kan det sägas att det lyckades med att identifiera vad som utgör auktionsdata och detta är då allt innehåll som finns i koden, taggad enligt följande:

[19] Network Working Grop: The application/json Media Type for JavaScript Object Notation (JSON). (Läst 2011-06-11.)
[20] Hypertext Preprocessor: *Home page*. (Läst 2011-06-12)

jcb({sessionsinfo, auktionsdata}).

4.2 Inhämtning

Problemet är följande, den nödvändiga informationen har blivit identifierad men hur kan en fristående applikation få tillgång till den informationen på ett automatiskt sätt utan inblandning av någon manuell process? Hela dataflödet ska också efterlikna, för en extern granskare, en helt vanlig person som sitter och ger bud via sin webbläsare. För att lösa detta används en tredjeparts programvara som i själva verket är ett tillägg till webbläsaren firefox, kallat iMacros[21]. iMacros är ett automatiserings-verktyg som, precis som namnet antyder, automatiserar aktiviteter som användare kan utföra i sin webbläsare. iMacros valdes för att det är gratis att använda som privatperson. Det har ett kraftfullt API samt en bra dokumentation för utvecklare och användare. Dokumentationen kommer både ifrån företaget som skapat produkten men också ifrån nätcommunityn som dagligen bidrar med hjälp för de som behöver det. För att iMacros ska hämta information från hemsidan så utnyttjas ett script som ser ut som följande:

```
"CODE:" +
                "VERSION BUILD=7211201" + "\r\n" +
                "TAB T=1" + "\r\n" +
                "TAB CLOSEALLOTHERS" + "\r\n" +
                "URL GOTO=http://***CENSURERAD***/" + "\r\n" +
                "'Use regular expression to extract data" + "\r\n" +
                "SEARCH SOURCE=REGEXP:\"jcb\\(([^;]+)\\)\" EXTRACT=$1" +
"\r\n" + "";
```

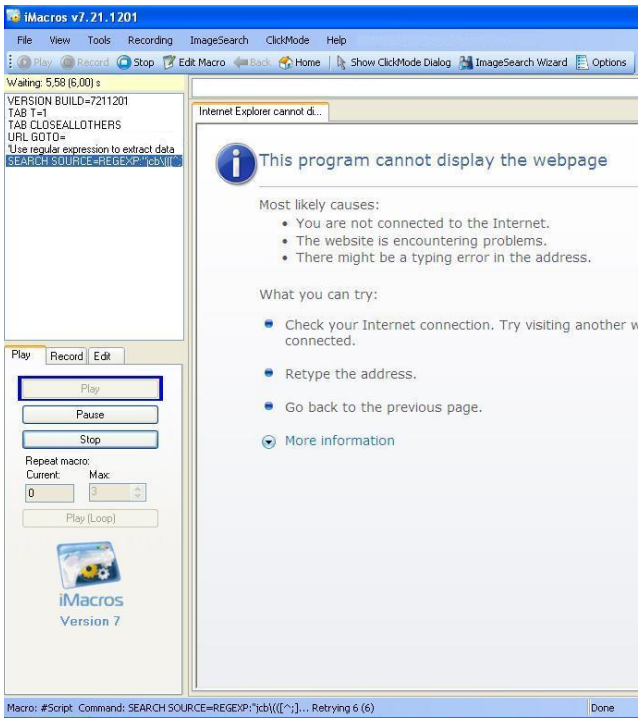
Detta fungerar på sådant vis att först kommer ett nyckelord ”CODE:” som tolkas som att efterföljande kod som kommer är ett script och ska köras därefter. Själva scriptkoden består av följande delar;

- iMacros versionsnummer.
- En flikhänvisning till vilken flik i webbläsaren koden exekveras i.
- En hänvisning att stänga ner alla andra flikar om sådana finns.
- En URL till vilken sida som ska laddas in.
- Ett regex (regular expression) som definierar start och slut för vilken data som ska sparas undan.

[21] IOPUS™. *iMacros.* (Läst 2011-06-12)

- En textvariabel, $1, där all data sparas undan.

Nu finns all auktionsdata sparat i iMacros som en stor kontinuerlig textsträng.



Figur 4:3 Automationsverktyget iMacros.

Bilden visar applikationen under körning.

Ovan bild visar hur iMacros försöker leta igenom en inte korrekt laddad sida med sitt regex sex gånger innan felmeddelande uppkommer. Ur bilden kan också avläsas hur en del av koden som skickades in till iMacros blev till kommentarer när iMacros tolkat det.

4.3 Konvertering

För att den extraherade informationen ska komma till nytta måste applikationen dela upp råtextsträngen som inhämtningen resulterat i, till individuella auktioner. Uppdelning sker genom att först lägga till en ny rad efter varje förekomst av teckenkombinationen ”:[”. Detta sker genom följande kodbit i c sharp där ”\r\n” är kombinationen för ny rad.

```
String rawTemp1 = rawExtract.Replace(":[{", ":[\r\n{");
```

Sen efter det så delas textsträngen upp ytterligare genom att ersätta kommatecken med ett nyradstecken i varje teckenkombination som lyder ”},{”. Detta sker i c sharp med hjälp av kodbiten:

```
String rawAuctionData = rawTemp1.Replace("},{", "}\r\n{");
```

Nu ser textsträngen, innehållandes all auktionsdata, ut såhär:

```
jcb({"st":"1297863895","pu":"0","hp":"0","ah":"c30eb733edf6b35dc602f5f249e0434c","ud":{},"au
ctions":[

{"ai":"20319","at":"2","st":"0","cs":"","sd":"1297713600","pi":"560","i":"30","ap":"IPAD","p
n":"Apple iPad","pl":"apple-ipad","pp":"4890","t":"1297863910","b":"49.78","u":"bsantes"}

{"ai":"20288","at":"2","st":"0","cs":"","sd":"1297796400","pi":"742","i":"30","ap":"CANONX",
"pn":"Canon EOS 60D + Objektiv","pl":"canon-eos-60d-
objektiv","pp":"10900","t":"1297863916","b":"16.86","u":"vccmv"}

{"ai":"20355","at":"2","st":"0","cs":"","sd":"1297850400","pi":"734","i":"105","ap":"PINK","
pn":"Dolce & Gabbana damklocka","pl":"dolce-gabbana-
damklocka","pp":"2000","t":"1297863994","b":"1.68","u":"dicosvd30"}

{"ai":"20357","at":"2","st":"0","cs":"","sd":"1297857600","pi":"309","i":"60","ap":"HM","pn"
:"H&M presentkort","pl":"hm-
presentkort","pp":"2000","t":"1297863921","b":"1.59","u":"Goodiie"}

{"ai":"20399","at":"2","st":"0","cs":"","sd":"1298228400","pi":"641","i":"30","ap":"SAMUE","
pn":"Samsung 46\" LED-TV","pl":"samsung-46-led-tv","pp":"11990","t":"29","b":"0","u":""}]});
            });
```

Texten har dock fått ett extra nyradstecken i denna framställning för att öka läsbarheten mellan varje individuell auktion. Nu är texten uppdelad och kan först nu användas i budauktionsprogramvaran för vidare bearbetning.

4.4 Test av applikationen

Testningen av applikationen genomfördes i två olika steg:

- Det första steget utfördes genom att under utvecklingen av datainhämtning testa applikationen direkt mot tjänsteleverantörens hemsida. Detta medförde att riktig auktionsdata kunde inhämtas och bearbetas.

- I den andra delen av utvecklingen, själva budgivningen, så testades aldrig funktionen för att buda mot tjänsteleverantörens skarpa hemsida. Detta för att undvika de juridiska påföljderna som annars skulle kunna vara möjliga. Däremot testades det en motsvarande funktion, att klicka på en knapp via iMacros, och detta utan problem.

Tjänsteleverantörens hemsida använder sig utav en databas för sparandet av data men hur den ser ut är en affärshemlighet. Av den anledningen och att det inte tjänar något syfte att i en testmiljö skapa en databas har inte en databas tagits fram. Detta för att i en testmiljö där enbart själva transaktionen av data till och från en server är viktig och inte hur data sparas på en server.

5 Applikationsutveckling av POC-applikation

Att utveckla mjukvara för automatisk budgivning över internet är en nödvändig del i detta arbete för att kunna bilda sig en rimlig uppfattning om

hur en sådan applikation skulle kunna fungera. Därför utvecklades en POC (Proof Of Concept) mjukvara för att förstå problemen men också om hur problemen skall kunna lösa. POC-mjukvaran utvecklades även för att praktiskt se om de tilltänkta skyddsåtgärderna mot programvara som hanterar automatisk budgivning verkligen fungerar. För att begränsa mjukvarans omfattning så har följande minimumkrav tagits fram för programvaran:

- Klara av att hämta alla bud som finns på den specificerade sidan.
- Klara av att presentera alla hämtade auktioner i ett gränssnitt.
- Begränsning av antalet bud som kan läggas.
- Begränsning av den maximala kostnaden för en vara.
- Inställning för hur ofta den specificerade sidan ska laddas om.

Programvaran kommer att byggas i språket C#[22,23,24] från Microsoft[25]. Detta språk valdes för att det är ett vanligt förekommande utvecklingsspråk bland många företag. Det valdes även för att det har inbyggt stöd för alla de funktioner som behövs, både i dagsläget men också i framtiden för programvaran. Det valdes tillika för att fördjupa och befästa undertecknads kunskaper inom språket.

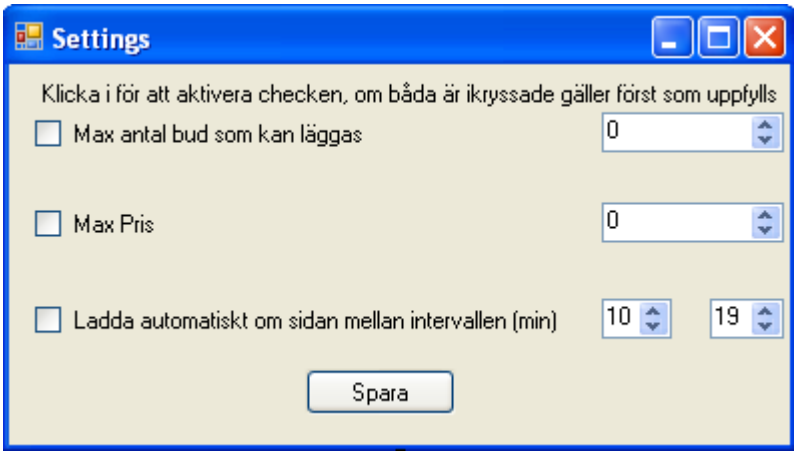
5.1 Inställningsmöjligheter

Antalet inställningar i applikationen är få men av betydande effekt. De tre inställningar som kan göras är följande:

- Begränsning av antalet bud som kan läggas.
- Begränsning av en varas maximal kostnad.
- Inställning för hur ofta den specificerade sidan ska laddas om.

Dessa inställningar representeras grafiskt enligt följande bild:

[22] Sharp, John. (2010). *Microsoft Visual C Sharp 2010 Step by Step.* Washington: Microsoft Press
[23] Dorman, Scott. (2010). *Teach Yourself Visual C Sharp 2010 in 24 Hours: Complete Starter Kit.* Indiana: Sams Publishing
[24] Schildt, Herbert. (2010). *C Sharp 4.0: The Complete Reference.* New York: McGraw-Hill Forouzan Networking Series
[25] Visual C# Devoloper Center. *The C# Language.*(Läst 2011-06-12)



Figur 5:1 Dialogrutan inställningar.

Bilden visar vilka inställningar som är möjliga i den egenutvecklade mjukvaran.

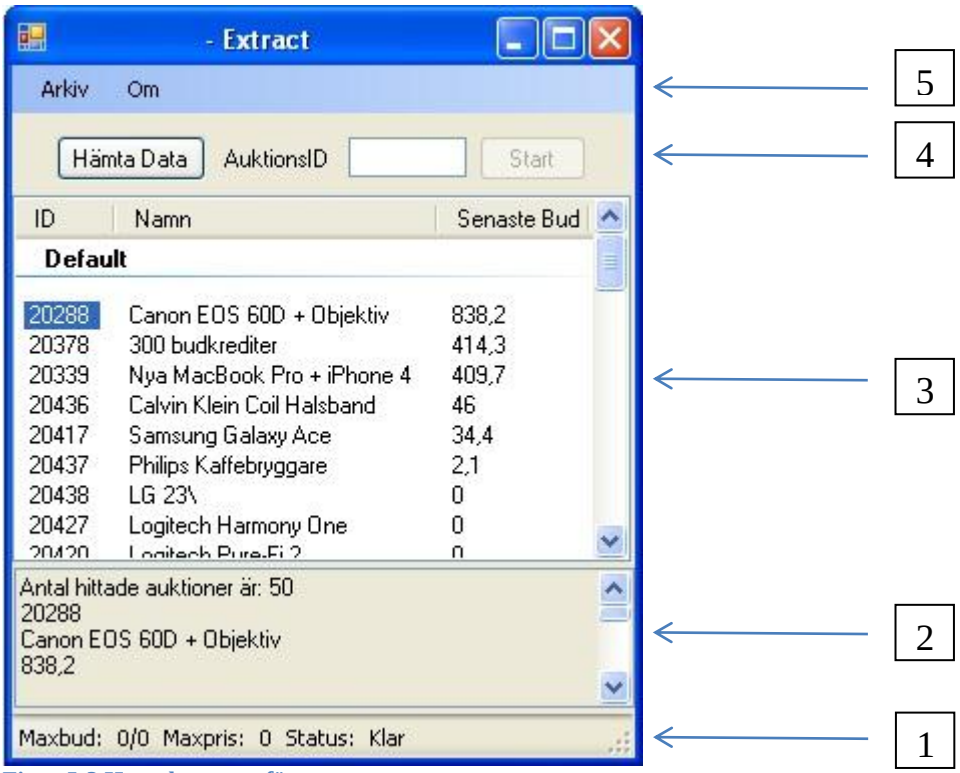
Inställningen för begränsning av antalet bud som kan läggas är för att förhindra att applikationen tömmer en användares budkrediter.

Inställningen för begränsning av en varas maximala pris är för att om varan överstiger angivet värde så stoppas autobudgivningen.

Inställningen för att automatiskt ladda om sidan mellan tidsintervallen är för att simulera en fortfarande aktiv användare för servern. Som aktiv användare räknas det om användaren antingen laddar om hela sidan eller lägger ett bud inom en tidsram på 20 minuter. Omladdning av sidan sker slumpvis mellan angivna värden.

5.2 GUI

Huvudprogrammet ser ut som följande bild:



Figur 5:2 Huvudprogramfönster.

Bilden visar hur det kan se ut när auktionsdata har blivit hämtat.

Det är uppdelat enligt följande:

1 - Statusbar som visar aktuella inställningar som användaren gjort.

2 - Loggfönster som skriver ut statusmeddelanden till användaren.

3 - Auktionsfönster, här presenteras alla tillgängliga auktioner för användaren.

4 - Aktivitetspanel, det är här applikationen gör det den är byggd för, flödet är enligt följande:

1. Användaren klickar på ”Hämta Data” knappen – Auktionsfönstret fylls då med tillgängliga auktioner.

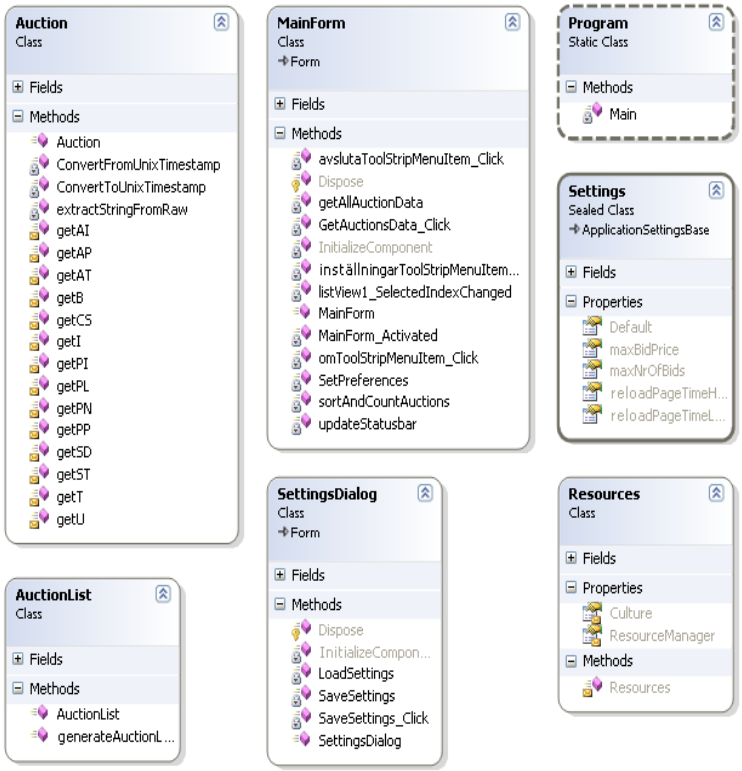
2. Användaren väljer en auktion i listan genom att dubbelklicka på den, alternativt att fylla i id manuellt i fältet AuktionsID.

3. Användaren klickar på Start. Applikationen är nu igång och sköter den automatiska budgivningen enligt användarens inställningar.

5 - Menyrad, ger användaren tillgång till inställningar och information om applikationen samt möjligheter till att avsluta applikationen.

5.3 Applikationssammansättning

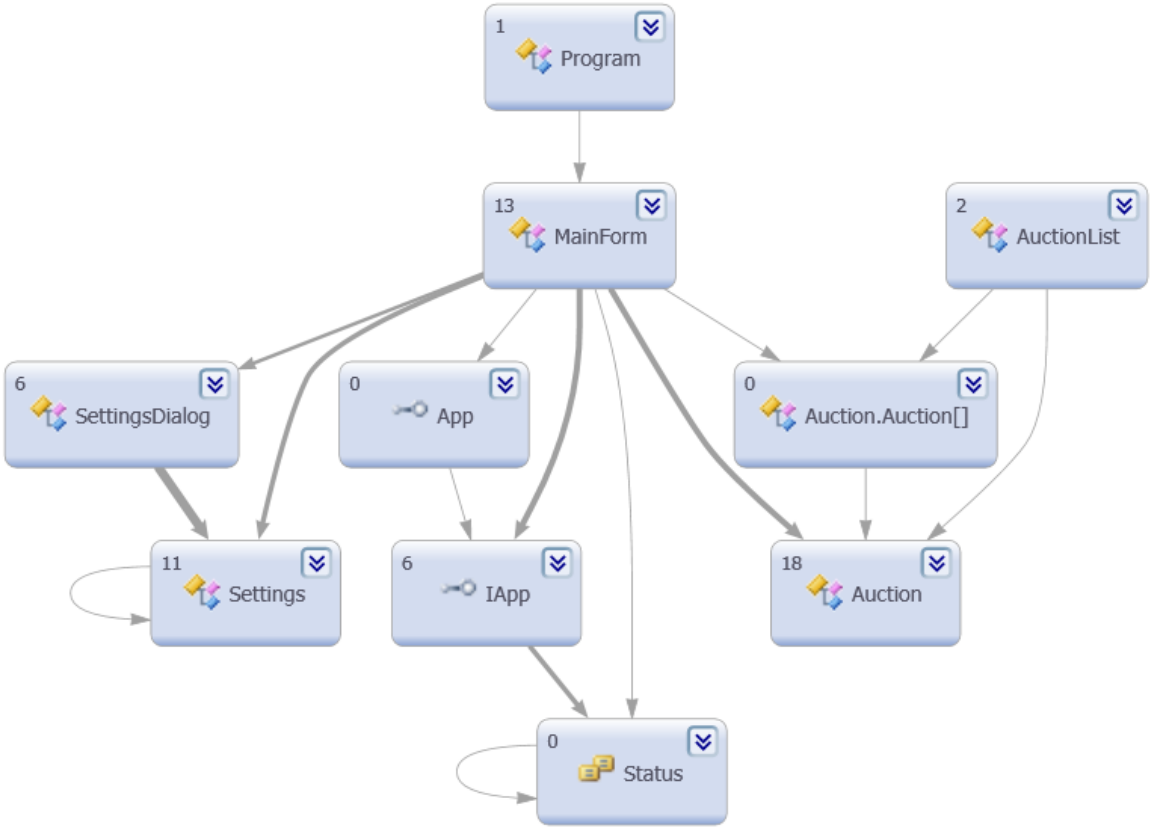
Applikationen har blivit uppdelad i olika klasser enligt följande överblick:



Figur 5:3 Klassöversikt.

Bilden visar samtliga klasser med metodnamn som ingår i den egenutvecklade mjukvaran.

Ett beroendeträd ser ut som följande:



Figur 5:4 Beroendeträd.

Bilden visar de olika beroenden som finns för den egenutvecklade mjukvaran.

En kort beskrivning av varje individuell klass behövs för att ytterligare öka förståelsen:

- Program – Är den övergripande klassen som innehåller Main metoden för att kunna köra applikationen.
- MainForm – Detta är huvudfönstret. Detta är vad användaren ser när applikationen startas. Härifrån nås alla menyval samt status för applikationen.
- SettingsDialog – Detta är fönstret som visar alla inställningar som går att göra i applikationen.
- Settings – Globala inställningar för hela projektet vid utveckling men också underliggande kod som anropas när användaren väljer en inställning i SettingsDialog.

- AuctionList – Tar emot osorterad rådata som innehåller auktionsdata och gör om det till en sorterad, lätthanterlig auktionslista som enkelt kan användas för presentation, filtrering och sökning.

- Auction – Är en samling operationer som går att utföra på varje individuell auktion som blivit skapad av AuctionList. Exempel på operationer är att hämta namn, starttid, budtid, budgivare.

- Resources – Autogenererad fil av Visual Studio 2010 vid projektstart som hanterar globala resurser. Användes inte i detta projekt.

6 Skyddsåtgärder

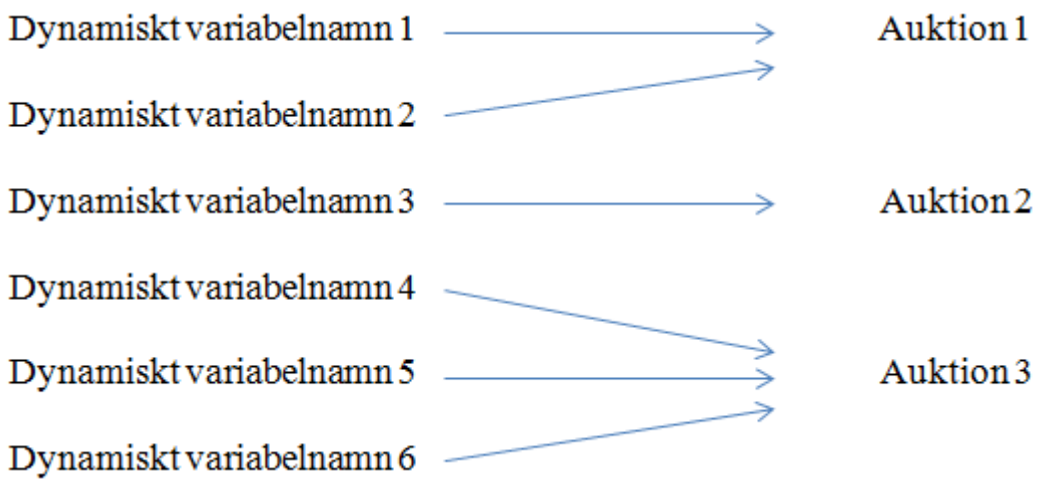
Efter att ha utvecklat en applikation som klarar av att automatisk buda på internetauktioner så ger detta upphov till frågan om hur det går att skydda sig mot sådana typer av applikationer. Det korta svaret är att det är svårt och att det ligger mycket i att de som tillhandahåller tjänsten är intresserade att förända sin webbplats så det blir svårare för den här typen av applikationer att fungera.

6.1 Dynamiska variabelnamn

Ett alternativ som skyddsåtgärd mot automatisk budgivning är att låta de variabelnamn som representerar auktionsdata vara dynamiska. Variabelnamnen bör vara unika för varje besökare och de skall bytas ut efter en viss tid, exempelvis var femte minut, trots att besökaren är kvar på sidan. Detta skulle effektivt förhindra all form av statisk identifiering av auktionsdata, vilket till exempel detta examensarbetes poc-kod bygger på. Tekniken skulle också kunna se till att den inbördes ordningsföljden för variablerna varierar. Med detta menas att vid en första sidladdning så genereras till exempel variablerna namn, auktionsid och pris. Vid en omladdning av sidan eller efter en förbestämd timer så ska en annan ordningsföljd erhållas.

Denna lösning har fördelen att den är transparent för slutanvändaren. Budgivare märker inte att auktionen som personen bjuder på just nu byter variabelnamn, till exempel från draenei till worgen eller något annat slumpvalt mönster av siffror och tecken.

Nackdelar med denna lösning är dock att hela auktionssidans bakomliggande system måste ha fullständig kontroll över vilket dynamiskt namn som är kopplat till vilken riktig auktion. En lösning för detta är ett buffertsystem som mappar alla dynamiska variabelnamn till det riktiga namnet. Bilden nedan illustrerar exemplet:



Figur 6:1 Dynamiska variabelnamn.

Bilden visar hur dynamiska variabelnamn är tänkt att kopplas till ett fast namn.

Implementationskostnaden för en sådan här lösning skulle bli högre om den kom i form av ett tillägg på en redan existerande lösning. Lösningen skulle inte bli dyrare om den byggdes in i ett auktionssystem från grunden.

6.2 Omdirigering vid inaktivitet

Att mäta tiden på hur länge en besökare varit aktiv i den form att besökaren inte har klickat på någon knapp, länk, bild eller laddat om sidan, så kan besökaren skickas vidare till en statisk sida som beskriver vad som hänt.

Fördelen med en sådan kontroll är att bandbredden hela tiden kan optimeras (sparas) för de besökare som faktiskt är aktiva på sidan. Detta innebär att de inte längre behöver hämta alla de förändringar i auktionerna som hela tiden sker. En annan klar fördel är att det är en mycket enkel åtgärd att implementera och som dessutom kan göras till ett lågt pris.

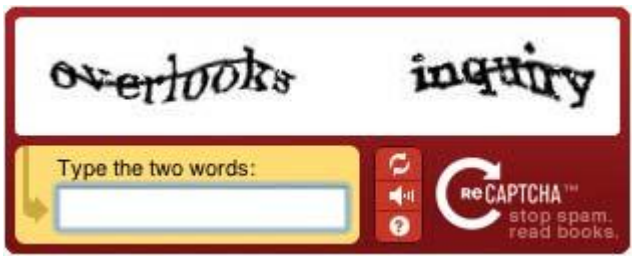
Nackdelen är dock att det är mycket lätt att ta sig förbi en sådan här kontroll genom att enkel simulera en aktivitet som då nollställer timern.

6.3 Captcha

Captcha är i sin enklaste form en bild av tecken, oftast bokstäver, som blivit förvridna eller manipulerade så att det är svårt för en dator att automatiskt avläsa dem, men fortfarande läsbara för en människa. Exempel på tre olika typer av captcha visas nedan:

Figur 6:2 Captcha av första generationen.

Bilden visar en captcha med enbart en lätt förvrängning av bokstäverna.



Figur 6:3 Captcha av andra generationen, alternativ ett.

Bilden visar en captcha med horisontala linjer igenom bokstäverna vilket ytterligare försvårar för automatisk igenkänning.[26]

Figur 6:4 Captcha av andra generationen, alternativ två.

Bilden visar en captcha där en ersättning för den horisontella linjen är att låta tecken överlappa varandra istället, detta används som ett alternativ/komplement till dem med horisontell linje.

Hur går det att dra nytta av captcha? Genom att implementera en captcha-kontroll på de auktioner som inte är tänkta att avslutas av tjänsteleverantörens egna automatiska budsystem. Då presenteras för vinnaren en captcha som måste besvaras rätt för att verifiera att vinnaren är en person och inte en budrobot. Denna kontroll skulle också kunna vara tidsbegränsad så att om inte rätt svar erhålls inom tidsramen så går vinsten automatiskt vidare. Denna händelse skulle också i sådana fall kunna registrera en varning internt i systemet för statistik och för att kunna se om en användare avviker från det normala.

Fördelarna med ett sådant system är att det är relativt enkelt och billigt att implementera i både nya och existerande lösningar. Det ger en förhöjd chans att stoppa helautomatiska verktyg men inte utan några nackdelar.

Nackdelarna med captcha är att personer som har något av följande kan komma att få problem trots att de är legitima användare:

- Blinda.
- Synskadade.

[26] Figur 6:3 Captcha - Encyclopædia Britannica Online Web. *Captcha: logo*. (Läst 2011-06-11)

- Färgblinda.
- Dyslektiker.

Nackdelen är också att trots captcha är aktiverat så kan en användare som befinner sig i närheten av en dator, eller kan fjärrstyra datorn som lägger det vinnande budet då enkelt ange rätt captcha och se ut som en helt legitim användare.

6.4 Verifieringskod via SMS/MMS/Email

En annan typ av skydd är att vid ett eventuellt vinnande bud ber om en kontrollkod som levereras via användarens förinställda nummer/email. Detta skulle då innebära att om användaren inte anger rätt kontrollkod inom en förutbestämd tidsram, så kan vinsten tillfalla näst högsta budgivaren, återgå till auktion eller vad tjänsteleverantören nu än beslutat.

Fördelen med denna typ av kontroll är att det inte skulle bli lika lätt att utveckla en automatiserad budgivningsrobot. Tjänsteleverantören skulle också ha en snabb och beprövad kommunikationskanal ut till användaren.

Nackdelarna är dock att tjänsteleverantören blir beroende av att användaren måste ange mer personliga uppgifter såsom email och/eller mobilnummer. Detta skulle kunna lösas genom att koppla sitt budgivningskonto till en extern SSO (singel sign on) leverantör med koppling till en social nätverksprofil, som till exempel Facebook. Då skulle användaren slippa ange mobilnummer i flera profiler för att detta skulle hämtas automatiskt via ens sociala profil. En ytterligare nackdel är att tjänsteleverantören gör sig beroende av en tredje kommunikationskanal, som vid ett eventuellt driftavbrott skulle innebära extra arbete för kunden då ett kontroll-sms/email inte skulle kunna nås. Omvänt gäller också att om inte tjänsteleverantören skulle mottaga rätt kontroll-kod så skulle detta skapa ett extra problem moment för den vinnande kunden. Fördröjningar kan också vara ett problem om dessa skulle uppstå och det innebär att tidsfönstret för att ange en korrekt kontroll-kod kan ha passerats.

6.5 Klientprogramvara

En applikation som utvecklas och tillhandahålls av tjänsteleverantören är en lösning där tjänsteleverantören bygger in alla funktioner som de vill ha. Tjänsteleverantören kan på detta sätt sätta upp regler som gäller för hur giltig kommunikation ska se ut på förhand. Exempelvis skulle en sådan regel innebära att enbart krypterad kommunikation på port 443/TCP (Https) är giltig. En annan fördel med denna typ av lösning är att själva applikationen kan kryptera/dekryptera all information så att även om någon avlyssnar trafiken så syns ingen data i klartext. Denna applikationskryptering skulle

kunna ske med en krypteringsalgoritm kallat RSA. Detta skulle då innebära att tjänsteleverantören kan koppla ihop ett användarkonto med en mjukvaruinstallation. Väljs istället en algoritm som utbyter nya nycklar vid varje ny sessionsinitiering, så är detta ett alternativ om inte det känns som om spårbarheten/nyckelhanteringen i RSA är tillräcklig.

En nackdel mot denna typ av lösning är att det kan bli problematiskt vid införandet av uppdateringar och utvecklingshanteringen då det kan förekomma många olika versioner som finns ute hos användarna.

6.6 Test av skyddsåtgärder

För att testa att de föreslagna skyddsåtgärderna verkligen fungerar i mer än teorin så utvecklades en testsida för ändamålet. Testsidan som är mycket enkel till sin utformning innehåller följande element:

- Auktionsdata – För att ge applikationen trovärdig data.
- En omdirigeringstimer – En räknare för passivitet.
- En auktionstimer – En räknare för en simulerad auktion.
- En bjudknapp till auktionen för en människa.
- En bjudknapp till auktionen för en robot.

Tyvärr är det så att testsidan skiljer sig källkodsmässigt ifrån en riktig tjänsteleverantörs hemsida. Detta har medfört att mjukvaran som utvecklats i detta arbete inte fungerar fullständigt mer än för att påvisa hur det skulle kunna se ut i en riktig situation. Av detta skäl så har enbart följande skydd kunnat testas rent praktiskt:

- Omdirigering vid inaktivitet (Se sektion 6.2).

De övriga skydden har enbart testats i teorin för att störa dataflödet i sådan utsträckning att automatiska budrobotar inte fungerar.

7 Framtiden

Att framtiden innehåller fler internetsidor som tillhandahåller tjänster som involverar auktions- eller budgivnings-tjänster är det ingen tvekan om. Ännu mer aktuellt blir det då för de som tillhandahåller tjänsten att se till så att fusk inte ska förekomma. Dels påverkar det deras varumärkes trovärdighet men

också kundernas tillit. Därför ser utvecklingen av hur tjänsteleverantörer ska kunna skydda sina tjänstesidor ljus ut.

7.1 Utveckling

Den mjukvara som togs fram under detta examensarbete var enbart tänkt som ett ”proof of concept”. Därför finns det mycket att önska på vad som kan implementeras i form av funktionalitet och prestanda.

7.1.1 Applikation

Den mjukvara som togs fram under detta arbetes gång bygger på användandet av tredjeparts mjukvara (iMacros). I en ”officiell” version så skulle det behöva byggas bort detta beroende för att optimera koden. En andra sak som också behöver göras är att bygga om grunden (grundfunktionaliteten) på sådant vis att applikationen utnyttjar sockets[27]. Detta för att kunna se all typ av trafik som flödar in och ut ur applikationen. Om detta införs så blir det betydligt enklare att kunna avläsa förändringsdata som skickas från webbservern till klienten. Detta skulle också resultera i en enhetlig mjukvara som kan fungera ändamålsmässigt utan stöd/beroende av tredje part. En funktion som också skulle vara mycket behändig skulle vara stöd för flera samtida budgivningar. Denna funktionalitet skulle verkligen påvisa styrkan i en automatisk budgivningsrobot.

7.1.2 Plattform

Smartphones eller ”smarta telefoner” har blivit mycket populärt och skulle kunna utgöra en alternativ plattform som komplement eller rentav ersättning för den traditionella mjukvaran på en pc. I sådana fall skall det också funderas över om mjukvaran skall byggas som en fristående server del som antingen körs lokalt på telefonen eller på en fjärr pc. I det sistnämnda fallet kan det utvecklas ett gränssnitt som fungerar på respektive plattform som administrerar serverdelen.

8 Slutsats

Automatiserad budgivning på internet är ett faktum. Det är först när detta är accepterat som tjänsteleverantörer kan utveckla sin tjänsteportal med denna tankegång som risken kan minskas för den här typen av fusk. Att ta bort risken helt är svårt, om inte omöjligt, det är därför en balansgång krävs mellan implementerade skyddsåtgärder och användarvänligheten. För mycket skydd och tjänsteleverantörerna riskerar att förlora kunder som då uppfattar processen med att lägga ett bud eller hämta ut en vinst som jobbig,

[27] Behrouz A. Forouzan. (2007). *Data Communications and Networking, Fourth Edition*. New York: McGraw-Hill Forouzan Networking Series (sid 706)

omständligt och/eller krävande. Motsatsen är om tjänsteleverantörerna har för lite skydd och de riskerar att bli mål för många automatiserade budgivningar.

Vilka skydd ska då satsas på? Personligen skulle jag vilja se att auktionssidor kan erbjuda följande föreslagna skyddsåtgärder:

- Omdirigering vid inaktivitet.
- Captcha.
- Verifierings-kod via SMS/MMS/Email.

Och i den prioritetsordningen. Detta är något alla tjänsteleverantörer skulle kunna implementera med den valfriheten att Verifierings-kod bör vara en valfri inställning hos användaren för sms om detta önskas utnyttjas.

Anledningen till varför jag valde ”Omdirigering vid inaktivitet” är för att det är det enskilt lättaste skyddet att implementera både i en ny och redan existerande lösning. Det ger också det mest grundläggande skyddet både mot den enklaste typen av robotar men också så ser denna lösning till att servern kan tillvarata bandbredden bättre.

Därefter kommer captcha och detta för att det är en av de skyddsåtgärder som kan ha bäst genomslagskraft för skydd av automatiserade budgivningsrobotar om de används på rätt sätt. Här skulle jag vilja se att tjänsteleverantörerna såg till så att en sådan finns med vid varje ny sessionsinitiering (både vid inloggning och efter inaktivitet) och vinnande bud.

Slutligen så har jag valt att ta med verifierings-kod via SMS/MMS/Email för att detta skulle vara ett bra komplement vid till exempel högvinst auktioner vars värde överstiger en viss summa. Detta skulle då ge tjänsteleverantören en extra nivå av skydd både mot budrobotar och mot kontokapningar, det vill säga där någons auktionskonto har blivit övertagit ofrivilligt av någon annan.

För att sammanfatta frågeställningarna i början som löd enligt följande:

- Vad är auktionsdata?
- Vilka data utgör en specifik auktion?
- Hur kan vi identifiera de olika delarna?

För att svara på den första frågan så utnyttjades Firebug och källkodsgranskning tillsammans för att ta reda på svaret. Resultatet var att all text som fanns mellan måsvingarna i källkoden som var inbäddade av jcb({data}) utgjorde auktionsdata. Detta leder oss till den andra frågeställningen som besvarades genom ytterligare granskning av den data som blivit identifierad. Här blev resultatet att varje individuell auktion utgjordes av den data som var innesluten i egna måsvingar och separerade med komma i det stora datablock som identifierades tidigare. Sista frågan besvarades genom att titta tillbaka på hur json ser ut och sedan applicera de reglerna på en individuell auktion för att identifiera de enskilda posterna. Den regel som tillämpades var att varje auktionspost hade en beskrivning och ett värde som var separerade med ett kolon. Sammanfattningsvis kan frågorna summeras enligt följande:

- All auktionsdata finns i jcb({all auktionsdata}).
- Varje individuell auktions data är inbäddad mellan måsvingar {individuell auktionsdata} och de olika auktionerna är separerade med ett kommatecken.
- Varje enskild post i varje auktion är bestående av ”beskrivning:värde” som är separerade med ett kommatecken.

Denna slutsats har då också besvarat de inledande fundamentala frågorna som löd:

- Går det att utveckla en applikation för automatiserad budgivning?
- Hur kan man skydda sig mot automatiserad budgivning?

Svaret på den första frågan kan sammanfattas med ett kort men kraftfullt ord: Ja. Svaret på den andra frågan kan mer sammanfattas som att det är möjligt att skydda sig om än enbart till en viss grad och då med bland annat tekniker som nämnts tidigare i detta arbete. Implementeringen av skyddstekniker är och kommer alltid att vara, en balansgång mellan användarvänlighet och säkerhet.

9 Källförteckning

9.1 Monografi

Behrouz A. Forouzan. (2007). *Data Communications and Networking, Fourth Edition*. New York: McGraw-Hill Forouzan Networking Series

Dorman, Scott. (2010). *Teach Yourself Visual C Sharp 2010 in 24 Hours: Complete Starter Kit.* Indiana: Sams Publishing

Schildt, Herbert. (2010). *C Sharp 4.0: The Complete Reference.* New York: McGraw-Hill Forouzan Networking Series

Sharp, John. (2010). *Microsoft Visual C Sharp 2010 Step by Step.* Washington: Microsoft Press

9.2 Webbdokument

Chris Pederick. *Web developer*. http://chrispederick.com/work/web-developer/. (Läst 2011-06-12)
Encyclopædia - Britannica Online Web. http://www.britannica.com/. (Läst 2011-06-11)

Firebug – Web devolopment enolved. *Home page*. http://getfirebug.com/. (Läst 2011-06-12)

GNU Operating System. (2010-09-08). *GNU General public license.* http://www.gnu.org/licenses/gpl-2.0.html. (Läst 2011-06-12.)

Hypertext Preprocessor. (2011-06-12). *General Information.* http://www.php.net/manual/en/faq.general.php. (Läst 2011-06-11.)

Hypertext Preprocessor. (2011-06-12). *History of PHP*. http://www.php.net/manual/en/history.php.php. (Läst 2011-06-11.)

Hypertext Preprocessor. (2011-06-12) *Home page*. http://www.php.net/ (Läst 2011-06-12)

Hypertext Preprocessor. (2011-06-12). *PHP Licensing.* http://www.php.net/license/. (Läst 2011-06-11.)

JavaScript Object Notation. *Introducing JSON*. http://json.org/. (Läst 2011-06-11.)

IOPUS™. *iMacros*. http://www.iopus.com/iMacros/. (Läst 2011-06-12)

Mozilla Firefox. *Firefox 4 är här!*. https://www.mozilla.com/sv-SE/firefox/ (Läst 2011-06-12)

Network Working Group. (2006-07-). *The application/json Media Type for JavaScript Object Notation (JSON)*. http://tools.ietf.org/html/rfc4627. (Läst 2011-06-11.)

Unicode 6.0.0. *Unicode 6.0.0.* http://www.unicode.org/versions/Unicode6.0.0/. (Läst 2011-06-11.)

Visual C# Devoloper Center. (2011). *The C# Language.* http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx. (Läst 2011-06-12)

Wireshark. *About Wireshark.* http://www.wireshark.org/about.html. (Läst 2011-06-12.)

Wireshark. *Home page.* https://www.wireshark.org/ (Läst 2011-06-12)

World Wide Web Consortium. (2011-06-11). *CSS Specifications.* http://www.w3.org/Style/CSS/current-work. (Läst 2011-06-11.)

World Wide Web Consortium. (2011-05-25). *HTML5 - A vocabulary and associated APIs for HTML and XHTML.* http://www.w3.org/TR/html5/. (Läst 2011-06-11.)

World Wide Web Consortium. (2011-04-26). *HTML/Element.* http://www.w3.org/wiki/HTML/Elements. (Läst 2011-06-11.)

World Wide Web Consortium. *Syntax and basic data types.* http://www.w3.org/TR/CSS21/syndata.html#q10. (Läst 2011-06-11.)

World Wide Web Consortium. (2011-03-14). *What is HTML.* http://www.w3.org/wiki/HTML/Training/What_is_HTML. (Läst 2011-06-11.)

9.3 Figurförteckning

Figur 6:3 Captcha:
Encyclopædia Britannica Online Web. (2011). *Captcha: logo.* http://www.britannica.com/EBchecked/media/124559/The-logo-for-CAPTCHA. (Läst 2011-06-10)

9.4 Tabellförteckning

10 Akronymer

API	Application Programming Interface.
CSS	Cascading Style Sheets.
GNU GPL	GNU's Not Unix General Public License.
HTTP	HyperText Transfer Protocol.
HTML	HyperText Markup Language.
JSON	JavaScript Object Notation.
MMS	Multimedia Messaging Service.
PHP	PHP: Hypertext Preprocessor.
POC	Proof of Concept.
RSA	Rivest, Shamir and Adleman.
SMS	Short Message Service.
SSO	Single sign-on.
TCP	Transmission Control Protocol.
USB	Universal Serial Bus.
XML	Extensible Markup Language.

11 Ordlista

Algoritm	Ett antal instruktioner för hur något ska genomföras.
API	En uppsättning regler för hur en viss programvara ska kommunicera med annan programvara.
Budkrediter	Valutan för hur många gånger det går att lägga ett bud.
Captcha	Teknik för att fånga upp (avslöja) automatiska program på internet.
CSS	Teknik för att beskriva hur layout ska se ut för ett dokument, till exempel storlek, färg och typsnitt.
Firebug	Tillägg för Firefox som kan analysera en hemsida.
GPL	Ett licensavtal för öppen källkod.
Hash funktion	En algoritm för att skapa ett, på förhand bestämt antal tecken, långt värde som är mindre än inmatningen. Detta värde ska också uppfylla kravet i kryptografiska sammanhang att det inte får vara möjligt att få ut klartexten om tillgång till hashen finns. Funktion av sådan typ kallas envägs-algoritm.
Http	Standard för hur online dokument ska överföras.
iMacros	En automatiserings programvara från företaget iOpus.
JSON	En standard för hur formaterad data ska överföras.
Objektorienterat	Programmeringsmetod för att kapsla in data i objekt.
PHP	Ett scriptspråk för inbäddning i html för skapandet av dynamiska hemsidor.
Protokoll	En uppsättning regler som beskriver hur en eller flera applikationer ska kommunicera med varandra.
Regular expression	Reguljärt uttryck som är en beskrivning på en metod för att hitta specifika delmängder ur en mängd. Till exempel vid sökning efter visst mönster i strängar.

RSA	En metod för att kryptera data med hjälp av publika nycklar.
Script	En typ av enklare instruktioner som slutanvändaren av en mjukvaruprodukt gör för att lägga till önskad funktionalitet.
Socket	En typ av adressering som innefattar både en IP-adress och ett port nummer.
SSO	Teknik för att auktorisera sig mot flera behörighetskontrollerande tjänster samtidigt.
TCP/IP	En fem lagers protokoll uppsättning som definierar hur utbyte av data sker över internet.
Visual Studio	En utvecklingsmiljö för programmering skapad av företaget Microsoft.
Web Developer	Tillägg för Firefox som kan analyser en websida.
Wireshark	En mjukvara för att analysera trafik i olika protokoll.

12 Index